Design and Testing of a Digital Diplexer for DOCSIS 4.0 Networks

A Thesis Submitted
to the College of Graduate and Postdoctoral Studies
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

Weicheng Wang

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 and Postdoctoral Studies 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 S7N 5A9
Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

The cable television industry has experienced massive growth since it began in the United States as a commercial business in 1950's. As cable television subscribers continued to grow in numbers, the demand for higher transmission rates increased. This led to the advent of Data-Over-Cable Service Interface Specification (DOCSIS). DOCSIS was developed by CableLabs and contributing companies to supervise the manufacturing of new digital equipment and ensure the compatibility of products from different manufacturers. New versions of the DOCSIS standards were released consecutively over the years to fulfill the raising demand for larger bandwidths and higher transmission rates. The latest version of the DOCSIS standards at the time of this writing is DOCSIS 4.0, in which the upstream and downstream signals are separated in frequency to eliminate interference.

A diplexer is a three-port device that implements this frequency-domain multiplexing, allowing bidirectional data transmission. It multiplexes two ports (e.g. L and H) onto a third port (e.g. S). The frequency bands occupied by the signals on ports L and H are separated by a transition band, which means that the signals on ports L and H can coexist on port S without interfering with each other. Commonly, the signals on port L will occupy a lower frequency band and the signals on port H will occupy a higher frequency band. In that case, a lowpass filter connecting ports L and S and a highpass filter connecting ports H and S are implemented in the diplexer. Conventional diplex filters have fixed passband and stopband corner frequencies, which means that they must be replaced every time there is a change in the bandwidth allocation due to customer demand. Furthermore, a conventional diplex filter usually has a large transition band due to the challenges of building a shard filter using radio frequency (RF) components, which results in wasted frequency spectrum in the network. One possible solution to the problems above is replacing conventional diplex filters with diplex filters built using digital hardware, which offers an unique advantage that allows filter parameters to be adjusted freely and precisely.

This thesis aims to design a hardware efficient digital diplexer for use in hybrid fiber-coaxial (HFC) networks. A digital diplexer samples the incoming analog signals, then per-

forms filtering digitally. The downstream and upstream sampling frequencies were optimized based on the DOCSIS 4.0 frequency division duplex spectrum options. The computation results showed that the ideal downstream sampling frequency is 3588 MHz, whereas the ideal upstream sampling frequency is 1616 MHz. Further, the frequency specifications of digital diplex filters were determined based on the frequency allocation defined by the DOCSIS standard. Multiple filter implementation structures were compared and contrasted to find a structure that supports high sampling frequencies at the lowest hardware cost. After careful consideration, block-based frequency domain filtering structure was selected and applied to the design.

Based on the filtering structure and parameters, a fixed point model of the digital diplexer was constructed in the Verilog hardware description language. A simulation was then conducted in ModelSim to verify the performance of the model in the FPGA development environment. Another fixed point model of the digital diplexer was built and tested in MATLAB. The testing results were evaluated and compared with the simulation results obtained in ModelSim, which aimed to verify the functionality of the designed diplexer. After that, more ModelSim and MATLAB simulations were conducted to verify that the designed diplexer achieves a signal quality that can support the highest modulation order specified in DOCSIS (4096-QAM) and allows for dynamic switching of the upstream/downstream transition point. In addition, several digital diplex filters with different sizes of transition band were designed and simulated in MATLAB. The results showed that digital diplex filters can achieve sharper transition bands and the 'wasted' bandwidth associated with higher split points can be reduced as compared to the conventional approach.

Acknowledgments

It is a pleasure to take this opportunity to express gratitude to those who made this thesis possible with their kindness and support.

First, I would like to thank my supervisor Dr. Brian Berscheid for helping me achieve the goal of completing this thesis and helping me throughout the research and writing process. It has truly been my honour and rewarding experience to work under his supervision. I would also like to thank Dr. Ha H. Nguyen for his guidance during the early stage of my research. A very special thanks goes to Dr. Eric Salt. He offered valuable information regarding the difference between masters programs available in University of Saskatchewan. I decided to pursue the degree of master of science after taking his advice.

My deepest love and gratitude go to my parents for supporting my studies with enthusiasm and encouragement. You have both allowed me to follow my dreams and have given me the opportunities needed to do so. My special thanks are extended to all my past and current research group members for sharing their knowledge and invaluable assistance.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

δ_p	Passband Ripple
δ_s	Stopband Attenuation
ω_p	Passband Corner Frequency
ω_s	Stopband Corner Frequency
f_c	Carrier Frequency
$H(e^{jw})$	Frequency Response
ADC	Analog-to-Digital Converter
ALM	Adaptive Logic Module
ALUT	Adaptive Look-Up Table
ASIC	Application-Specific Integrated Circuit
ASK	Amplitude-Shift Keying
AWGN	Additive White Gaussian Noise
CATV	Cable Television
CM	Cable Modem
CMTS	Cable Modem Termination System
CP	Cyclic Prefix
DAC	Digital-to-Analog Converter
dB	Decibel
DFT	Discrete Fourier Transform
DOCSIS	Data-Over-Cable Service Interface Specification
DS	Downstream
DSL	Digital Subscriber Line
DSP	Digital Signal Processing
FDD	Frequency-Division Duplexing
FDM	Frequency-Division Multiplexing
FDX	Full Duplex

FFA	Fast Finite Impulse Response Algorithm
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HDL	Hardware Description Language
HFC	Hybrid Fiber-Coaxial
HPF	Highpass Filter
IDFT	Inverse Discrete Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
IP	Intellectual Property
ISBE	International Society of Broadband Experts
ISI	Inter-Symbol Interference
LE	Logic Element
LNA	Low Noise Amplifier
LPF	Lowpass Filter
LSB	Least Significant Bit
MSB	Most Significant Bit
MSE	Mean Squared Error
NCTA	National Cable and Telecommunications Association
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
PA	Power Amplifier
PHY	Physical Layer
QAM	Quadrature Amplitude Modulation
QoS	Quality-of-Service

RAM	Random-Access Memory
RF	Radio Frequency
S-CDMA	Synchronous Code-Division Multiple Access
SC-QAM	Single-Carrier Quadrature Amplitude Modulation
SCTE	Society of Cable Telecommunications Engineers
SNR	Signal-to-Noise Ratio
SOPC	System-On-a-Programmable-Chip
US	Upstream
VHSIC-HDL	Very High Speed Integrated Circuit Hardware Description Language

1. Introduction

1.1 History of Cable Television

Cable television (CATV) originated in the United States in the late 1940's and early 1950's, a time when few television stations were operating and could only be found in major urban areas. The main purpose of CATV was to enhance poor reception of over-the-air television signals in rural areas due to geographic factors. Innovators in Arkansas, Oregon and Pennsylvania independently created community antenna/access television systems [1]. An antenna tower would be erected on mountain tops or other elevated points in the community to capture television signals broadcast from neighboring major cities, and homes were connected to the antenna tower via coaxial cables, where amplifiers were placed properly to compensate for the attenuation of the cables.

The demand for CATV service continued to grow in rural America. By 1960's, there were approximately 800 cable systems in operation, which provided services to almost 850,000 subscribers. Large corporations such as Westinghouse, TelePrompTer and Cox started putting investments into the business [1]. The early economic structure of CATV showed clear evidence that the birth and evolution of CATV was driven by custore demand [2]. The drastic growth of CATV was also caused by the introduction of several technological innovations over the decade. One of the most important innovations was the addition of upstream channel that makes two-way communication possible. Paul Baran, founder of Arpanet and analyst of RAND Corporation, made a significant contribution to the development of the concept [3]. As a result, the market of CATV expanded beyond rural areas and reached saturation by the late 1960's.

The number of proposed interactive services for CATV systems was very large. E. K.

Smith created four main categories of interactive services that were explored during the upstream prototyping phase in 1970's [4]:

1. "narrow-band subscriber response services (e.g. opinion polling, sensor monitoring, pay-TV);"
2. "shared two-way channels (e.g. local ombudsman, remote medical diagnosis, neighborhood program origination);"
3. "subscriber-initiated services (e.g. computer time sharing, reservation and banking services, catalogue shopping);"
4. "point-to-point services (e.g. high-speed data-exchange, teleconferencing, and facsimile)."

Only a few services were offered before the introduction of digital cable years later, among which pay-TV was recognized as the most popular service, which motivated most CATV service providers and manufacturers to shift their focus towards developing effective pay-TV networks.

The advent of the internet in the late 1980's and early 1990's changed the cable industry dramatically. The cable industry began making the transition from analog to digital signal distribution due to the rising demand for more efficient services. This transition involved replacing the traditional distribution networks with hybrid fiber-coaxial (HFC) networks, which is a broadband network that combines fiber optic and coaxial cable. The HFC networks introduced faster and higher quality data transmission to customers, which allowed it to compete with digital subscriber line (DSL) technology provided by many telephone companies [5]. The HFC networks can also offer multichannel video, two-way voice, and high definition and advanced digital video services [1].

Initially, there were no industry-wide standards in place to manage the cable equipment demanded by the CATV service providers. The absence of standards resulted in the service providers being forced to purchase proprietary equipment at whatever price demanded by

the manufacturer. The other issue caused by the lack of standards was the incompatibility of the equipment manufactured by rival companies, which forced the service providers to trade with a particular company. As a result, the service providers collaborated to form CableLabs, a not-for-profit organization that generated open standards to ensure the compatibility of cable equipment between various equipment manufacturers. The most important standard developed by CableLabs was the data-over-cable service interface specification, DOCSIS.

1.2 History of DOCSIS

By 1997 CableLabs released the first data-over-cable service interface specification standard, DOCSIS 1.0. The physical network defined by DOCSIS was comprised of multiple cable modems (CMs) connected to a cable modem termination system (CMTS) over an HFC network [6], as shown in Figure 1.1. The CMTS is locted at the service provider's headend and connects to many optical nodes via fiber optic cables. The optical signal from the CMTS is converted into radio frequency (RF) and transmitted to multiple taps via copper coaxial cables [7]. Each tap delivers a portion of the signal power on the cables to the CMs at users' locations.

A DOCSIS network enables two-way transmission. The transmission from the CMTS to the CMs through the HFC network is called forward path or downstream (DS) transmission, and return path or upstream (US) transmission refers to the transmission from the CMs to the CMTS. There were limited changes to the physical structure of DOCSIS network despite the fact that CableLabs has released new versions of DOCSIS standard over the years. The updates to the DOCSIS standard typically involve improving both upstream and downstream data rates. These updates have included versions 2.0, 3.0, 3.1 and 4.0, which was released in 2019 and is the latest version. It is worth mentioning that the transmission rate of fiber optic cable is higher than that of copper coaxial cable, which means the copper coaxial cable is the bottleneck in the network. Therefore, updates to the DOCSIS standard aim to increase the transmission rate over copper coaxial cable through updates to the physical signaling and medium access control protocol. Table 1.1 compares several aspects of each DOCSIS version, including their maximum data rates.

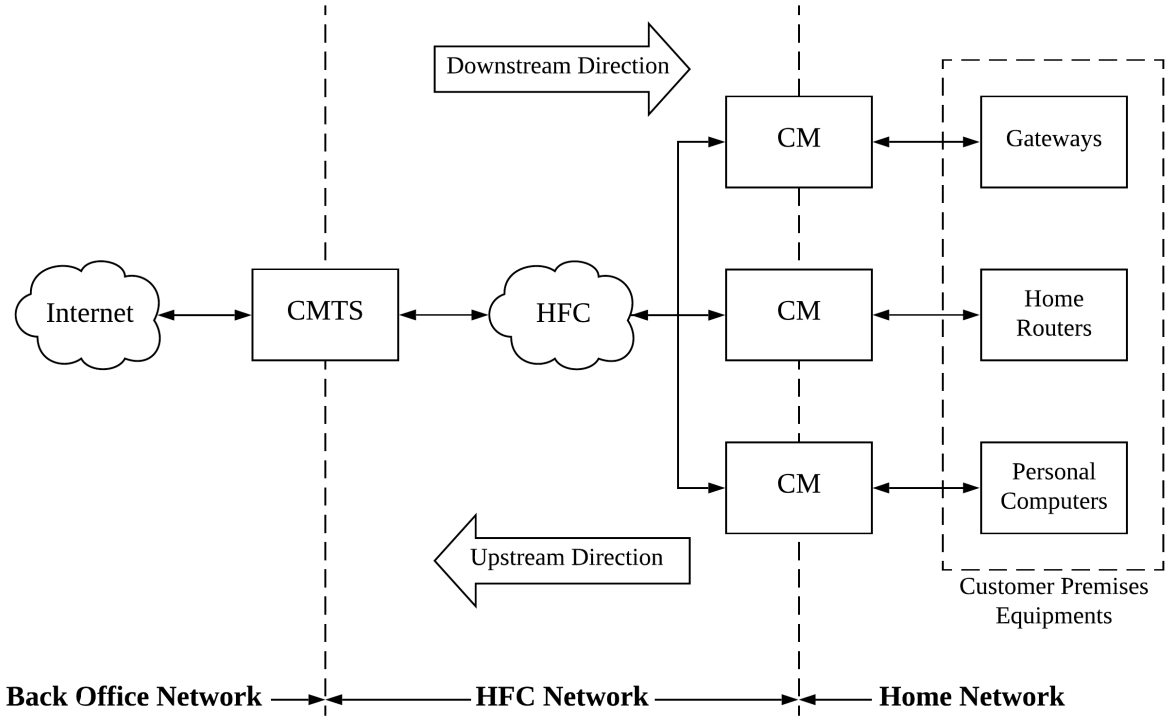


Figure 1.1: The DOCSIS network

Table 1.1: The evolution of DOCSIS

Specification	DOCSIS 1.0 [6]	DOCSIS 1.1 [8]	DOCSIS 2.0 [9]	DOCSIS 3.0 [10]	DOCSIS 3.1 [11]	DOCSIS 4.0 [12]
Release Date	1996	1999	2001	2006	2013	2019
Downstream Capacity	40 Mbps	40 Mbps	40 Mbps	1 Gbps	10 Gbps	10 Gbps
Upstream Capacity	10 Mbps	10 Mbps	30 Mbps	200 Mbps	1-2 Gbps	6 Gbps
Multiplexing Technique	FDMA, TDMA	FDMA, TDMA	FDMA, TDMA, S-CDMA	FDMA, TDMA, S-CDMA	OFDM, TDMA, S-CDMA	OFDM, TDMA, S-CDMA

1.3 DOCSIS 4.0

DOCSIS 4.0 technology introduced the next generation of data-over-cable's hybrid-fiber coaxial networks. It allowed for multi-gigabit symmetric services while supporting high reliability, high security and low latency. DOCSIS 4.0 included many significant enhancements over DOCSIS 3.1. Most notably, the maximum upstream capacity increased from 1.5 Gbps [11] to 6 Gbps [12]. The maximum downstream capacity remained unchanged, which is 10 Gbps [12]. There was a considerable increase in the bandwidths allocated to transmission in the upstream and downstream directions as well. The downstream bandwidth could span from 108 MHz to 1794 MHz, and the upstream bandwidth traditionally spans from 5 MHz to 85 MHz [12]. Many future applications that are currently under development can benefit from higher upstream transmission rates, which include interactive video conferencing, remote learning and health care applications.

The implementation of OFDM and OFDMA has made an important contribution to the sharp increase in data rates in DOCSIS 3.1 and 4.0. OFDM is a flexible and efficient modulation technique that transmits over a large number of closely spaced orthogonal subcarriers in parallel at low symbol rate instead of delivering a high-rate data stream with a single subcarrier.

In an OFDM system, the input data stream is modulated by the serial-to-parallel converter and distributed amongst the subcarriers. The QAM mapper then maps information bit from each active subcarrier to a QAM constellation, resulting in a complex symbol stream. Following mapping, the symbol stream is converted to its time domain equivalent by taking an inverse fast Fourier transform (IFFT). Next, a very important operation called cyclic prefix (CP) extension is performed on the complex symbol at the output of IFFT. This cyclic prefix can effectively eliminate inter-symbol interference (ISI) between the complex symbols, given that it has a greater length than the channel delay spread.

The upstream transmission is built based on the OFDMA digital modulation scheme. OFDMA is a multi-user version of OFDM that assigns subcarriers to various users on a frame-by-frame basis, allowing for simultaneous transmission from multiple users at low

data rate.

The potential for the upstream and downstream signals to interfere with each other in DOCSIS networks is high due to their coexistence on the same coaxial cable. This problem is traditionally solved in DOCSIS by utilizing a scheme called frequency-division duplexing (FDD), where the upstream and downstream signals are allocated to non-overlapping frequency bands. Wherease earlier versions of DOCSIS specified fixed upstream and downstream frequencies, DOCSIS 4.0 introduced a new mode which allows the upstream band to extend to 300 MHz, 492 MHz, and 684 MHz. A summary of the various frequency band options allowed in DOCSIS 4.0 is demonstrated in Figure 1.2. Since the achievable data rate in a network is related to the allocated bandwidth, this mode gives cable network operators the ability to balance their upstream and downstream network capacity based on actual customer demand.

One of the key elements in an HFC network is the optical node, which acts as the interface between the optical and electrical portions of the network. An optical node consists of a optical receiver that recovers and then transmits the downstream signal from the CMTS to the CMs, and a reverse optical transmitter that delivers the upstream signals from the CMs back to the CMTS. Figure 1.3 demonstrates how optical nodes are utilized in an HFC network, whereas the basic structure of an optical node is shown in Figure 1.4. The physical layer (PHY) device is utilized to set up communication with the CMTS at the service provider's headend. The downstream PHY receives the downstream optical signal from the CMTS and converts it to a radio frequency signal that is transmitted to the CMs via a coaxial cable [13]. The digital-to-analog converter (DAC) ensures that the RF signal at the output of the optical node is analog. Before the RF signal enters the coaxial cable, a tilt in the frequency domain must be added to the RF signal to compensate for the attenuation of coaxial cable network. The tilt is opposite to the slope of the attenuation in the cable network. Following the tilt compensator, a power amplifier (PA) is implemented to boost the power in the transmitted signal so the signal has enough power to reach the CMs located farthest away on the coaxial cable network.

The path of the upstream signal is adjacent to that of the downstream signal. As il-

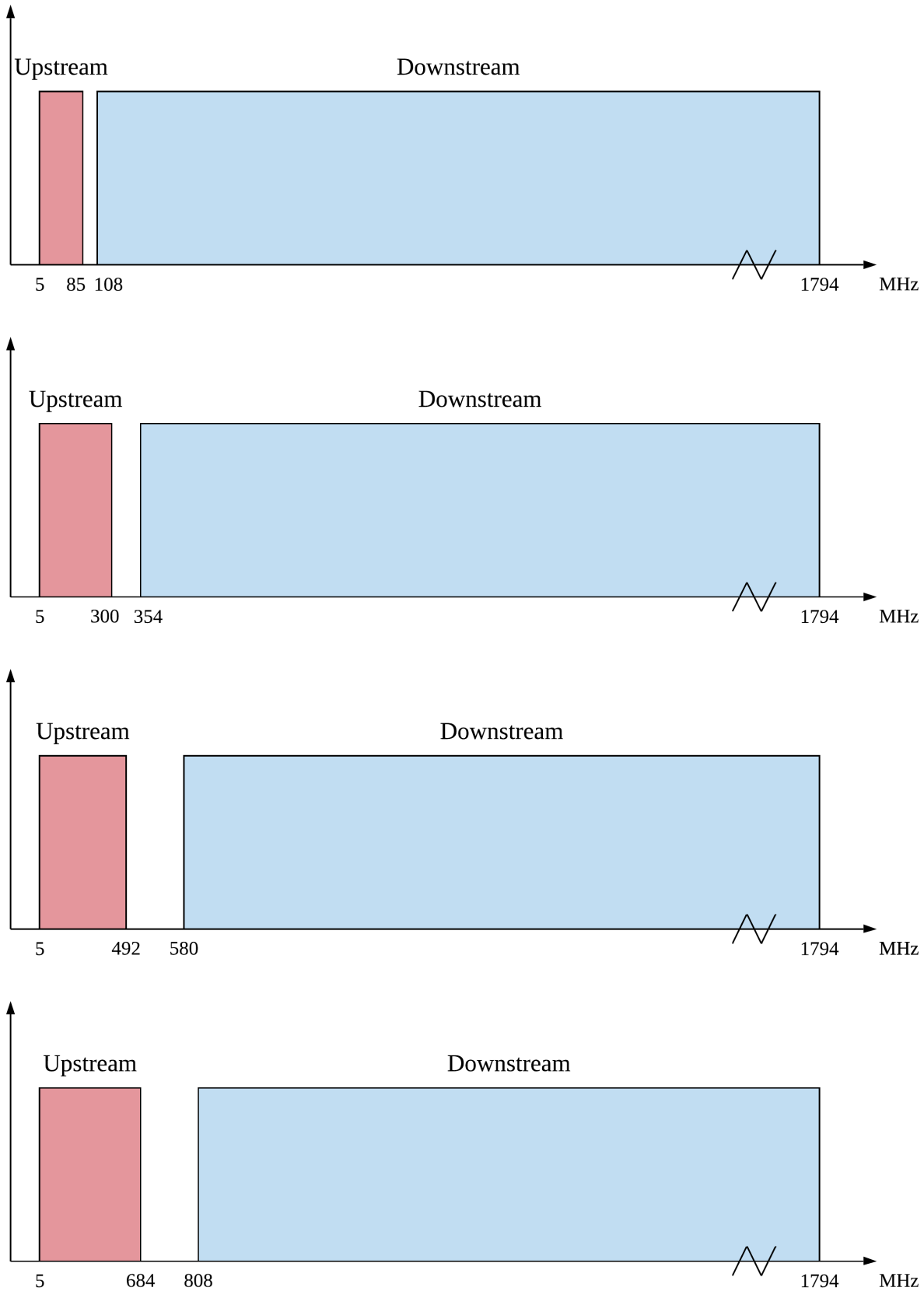


Figure 1.2: DOCSIS 4.0 frequency division duplex spectrum options

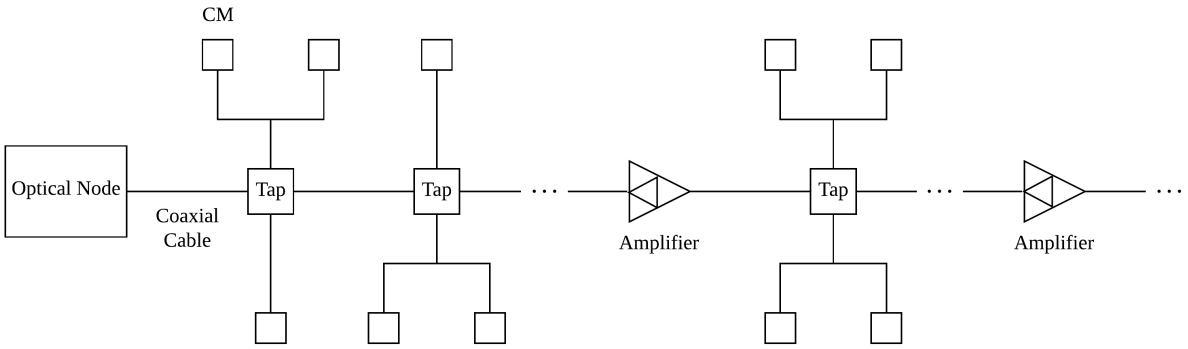


Figure 1.3: Optical node and amplifiers in an HFC network

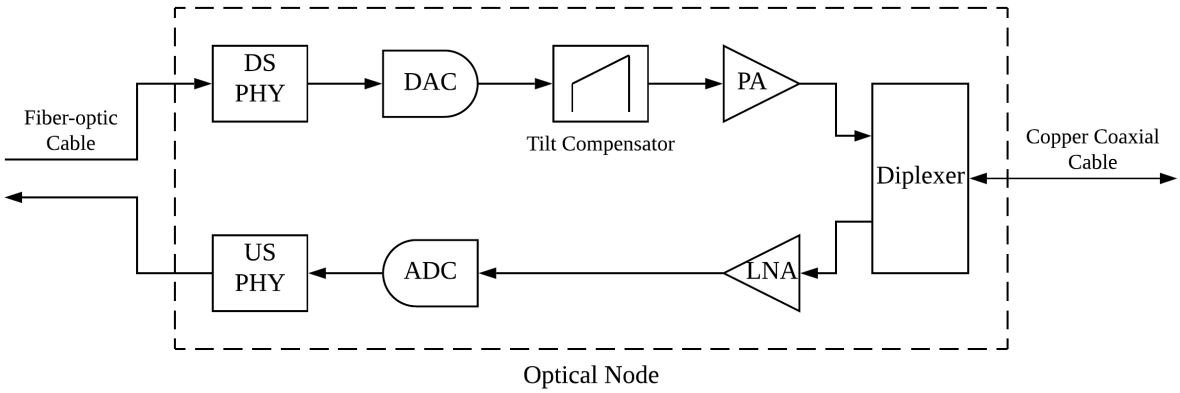


Figure 1.4: DOCSIS 4.0 optical node interface

lustrated in Figure 1.4, a low noise amplifier (LNA) is used to increase the power in the upstream signal sent by the CM. The signal is then sampled by the analog-to-digital converter (ADC). It becomes an optical signal after being demodulated by the upstream PHY and travels to the CMTS over a fiber-optic cable.

To facilitate the bidirectional operation of the optical node, a device known as a diplexer is used to separate the downstream and upstream signals coexisting on the same coaxial cable, as shown in Figure 1.4. A diplexer is a three-port network that implements frequency-domain duplexing. As discussed earlier, in a DOCSIS network, the upstream signal will occupy a lower frequency band and the downstream signal will occupy a higher frequency band. In that case, the diplexer functions as a lowpass filter (LPF) for the upstream signal,

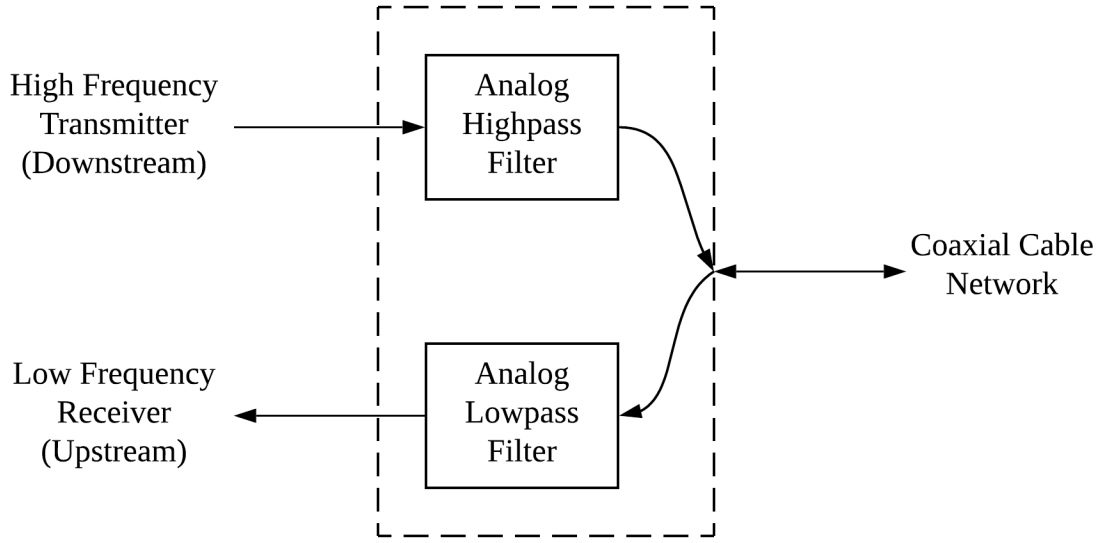


Figure 1.5: Basic concept of a traditional diplexer

and serves as a highpass filter (HPF) for the downstream signal, essentially separating the signals into separate upstream and downstream paths within the optical note, as shown as shown in Figure 1.5 [14]. It is important to note that the lowpass and highpass filters discussed above are collectively referred to as diplex filters, which are designed to eliminate or receive signals that fall in certain areas of the RF spectrum. The purpose of the diplexer is to prevent the downstream and upstream signals from interfering each other. The insertion loss between the input and output ports of the diplex filter is approximately 0.5 dB [15].

As introduced in Figure 1.3, amplifiers are another important component of HFC networks. The electrical signals that travel between the optical node and the CMs are attenuated as they propagate down the coaxial cable [16]. At a certain point, their signal level becomes too low for reliable communication. In order to extend the signal reach from the optical node, amplifiers are placed periodically along the cable. It is important to note that due to the bidirectional frequency division duplexing nature of the network, the amplifier is not merely a simple device that boosts the level of an input signal, as studied in introductory electronics courses. Rather, the amplifiers in HFC networks utilize a diplexer to first separate the upstream and downstream signals, invididually amplify the two signal paths, then apply

another diplexer to recombine the signals onto a single coaxial cable. Once again, it is seen that the construction and operation of the HFC network relies upon the use of diplexers to manage the bidirectional signals.

1.4 Problem Statement

HFC networks have experienced rapid growth since their inception in the early 1990s. Network operators have continually introduced technological innovations and provided ever-increasing bandwidth capacities to fulfill the user demand for higher upstream and downstream transmission rates. To increase bandwidth capacities under DOCSIS 4.0, the upstream and downstream frequency bands can be reallocated on the frequency spectrum. However, doing so means the filters inside the diplexers need to be modified to match the updated frequency spectrum. Unfortunately, traditional diplex filters are built from fixed RF components (resistors, capacitors, inductors, etc.) and therefore have fixed passband and stopband cutoff frequencies. This means that network service providers must replace every diplexer in the HFC network when changing the way frequencies are partitioned between the upstream or downstream direction. This upgrade process can be very expensive when conducted across an entire country or continent [17].

Furthermore, the traditional diplex filters require a relatively large guard band between the upstream and downstream frequency bands to accommodate the filter transition region. As already mentioned in the previous section, diplex filters are designed to transmit or receive RF signals. Due to the limitations of analog RF components, the size of the required transition band increases as the upstream or downstream frequency band becomes larger. This can be seen in Figure 1.2, where the transition band in for an 85MHz DOCSIS upstream is 23 MHz, while the transition band in a 684MHz DOCSIS upstream is 124 MHz. In other words, increasing the upstream bandwidth to meet customer demand for interactive applications would cause the transition band to be increased, wasting a huge portion of the frequency spectrum since no signals may fall in the transition region.

DOCSIS 4.0 specification introduced a technology called Full-Duplex DOCSIS (FDX) that could aimed to solve the problems above. This technology enables upstream and down-

stream transmissions to occur in the same frequency band at the same time. The advantage of utilizing FDX is that it potentially allows upstream bandwidth expansions to occur without reducing downstream bandwidth commensurately. FDX also removes the transition band between the upstream and downstream frequencies, thereby making full use of the frequency spectrum. In addition, the overlapping frequency bands can occupy a variety of ranges: 108 204 MHz, 108 300 MHz, 108 396 MHz, 108 492 MHz, or 108 684 MHz [17], allowing for flexible upstream and downstream band edges. Unfortunately, FDX solutions do not currently work with amplifiers. Although some efforts are underway to combine FDX operation with amplifiers, the current solutions require fairly complex echo cancellation schemes [18] [19]. As such, FDX may only be useable by the CATV service providers that are willing to spend significant time and money replacing all of the amplifiers in their networks with optical nodes.

Another an alternative method of solving the problems associated with conventional diplexers and unlocking the potential of DOCSIS 4.0 is to design a flexible and dynamically configurable diplexer. This thesis investigates the possibility of designing such a device based on digital filtering, rather than analog filtering using RF components. The physical structure of a digital diplexer will be discussed in detail in later sections of the thesis, but involves sampling the incoming analog signals, performing digital filtering, then finally converting the digital signals back into analog and placing them on the coaxial cable. The utilization of digital filters offers a powerful benefit to this appilcation, in that the filter's response can be programmable or adaptive [20]. In other words, digital filters can have flexible cut-off frequencies, allowing for adjustable upstream and downstream bandwidths. The size of transition band is also adjustable for the same reason.

Such a diplexer based on digital filtering could be included within the optical node and the amplifiers within DOCSIS networks, providing the opportunity for the network to dynamically allocate frequency resources between the upstream and downstream paths. However, there are important challenges to be solved. First, since the signals of interest have wide bandwidths, the sampling rates, computational complexity, and hardware costs of the digital diplexer could be very high. Second, since DOCSIS networks aim to provide very high data

rates, the signal quality must not be degraded by the diplexing operation. Thus, one of the key objectives of this thesis is to design a functional digital diplexer that can support DOCSIS 4.0 data rates at the minimum hardware cost.

1.5 Thesis Outline

This thesis is organized as follows. Chapter 2 will provide an overview of digital filters and briefly explain why digital filters are preferred over analog filters. The digital filter design requirements for this application will be discussed in two aspects: sampling rate and signal-to-noise ratio. In addition, there will be a detailed discussion of the underlying QAM and OFDM systems used in DOCSIS in Chapter 2.

Chapter 3 will consider various filtering structures for implementing the digital diplexer and evaluate their performance and hardware costs. The advantages and disadvantages of these structures will be discussed, which will ultimately lead to the selection of a filtering structure for the diplexer.

Chapter 4 will explain how to implement the proposed digital diplexer from Chapter 2 and 3 and will investigate its practical limitations. The general implementation and testing procedure will be discussed in detail, and the performance of the newly developed system will be evaluated based on simulation results.

Finally, Chapter 5 will conclude the thesis by summarizing the main developments and discussing potential future work.

2. Filter Design Requirements

This section first provides an overview of the fundamentals of digital filters, then discusses the key concepts of QAM modulation and OFDM communications as used in DOCSIS 4.0. Finally, this material is used to develop key requirements for the filters used to construct the digital diplexer.

2.1 Digital Filter Overview

A digital filter is a discrete-time system which takes an input signal and produces an output signal that is a modified version of the input signal. In situations where the signal to be processed is an analog signal (as in the current application), the implementation of a digital filter is generally comprised of an analog-to-digital converter, a digital logic circuit such as a microprocessor, peripheral components such as memory, and a digital-to-analog converter (DAC). The analog-to-digital converter (ADC) serves to sample the input signal, while filter coefficients and data are stored in the memory. The programmed instructions running on the microprocessor enable it to perform mathematical operations on the signal obtained by the ADC. Finally, the filtered signal is converted back into the analog domain using the DAC. In some high-performance applications which require high sampling rates or multiple parallel operations, a field programmable gate array (FPGA) or application-specific integrated circuit (ASIC) is preferred over a general-purpose microprocessor. The digital logic device that was used in this thesis is an FPGA, which is a semiconductor device consisting of programmable logic elements (LEs) as well as reconfigurable interconnections which can be used to produce any types of complex logic circuits [21]. Implementation of a digital filter in a practical situation typically requires using software or mathematical

optimization routines to determine appropriate filter coefficients based on the parameters defined by users [20].

Digital filters are different from analog filters in many ways. First, a digital filter is typically less efficient in size and power, while being less component-sensitive, than its analog counterpart. Generally, the disparity in efficiency rises as the frequency of signal increases, as the digital logic needs to operate at higher sampling rates. Secondly, the hardware cost of a digital filter may be higher than that of an equivalent analog filter due to the increased complexity. However, digital filters allow many designs that are impractical or impossible as analog filters, as mathematical operations can be performed at very high precision. Lastly, digital filters occasionally have problematic latency (the difference in time between the input and the response) when being used in the circumstances of real-time analog systems, which is due to the associated analog-to-digital and digital-to-analog conversions and logic delays associated with the implementation in the digital circuit.

A digital filter is usually defined by its transfer function, or equivalently, its difference equation. A transfer function of a filter is a mathematical function that generates the theoretical output signal for each possible input signal. The filter response to any given input can be modeled through mathematical analysis on the transfer function of the filter. Thus, designing a digital filter requires generating specifications based on the problem definition, and then building a transfer function that fulfills the specifications. The transfer function of a linear, causal, and time-invariant digital filter can be demonstrated as a function in the z-transform domain [22]:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_M z^{-M}} \tag{2.1}$$

where b_n and a_n are known as the coefficients of the filter. The filter order of the digital filter is defined as the greater of N and M in the above equation.

There are various mathematical methods that can be utilized to design a digital filter and simulate its performance. These methods often help developing the basis of a filter specification. Generally, a digital filter can be characterized through computing its response to a simple input such as an impulse, which can then be extended to calculate the filter

response to more complicated signals.

In the time domain, a digital filter is entirely characterized by the impulse response $h[n]$, which is the measurement of the filter output in response to an impulse. Generally, there are two basic types of digital filters: finite impulse response (FIR) and infinite impulse response (IIR) filters. The impulse response completely matches the sequence of filter coefficients in the case of a linear time-invariant FIR filter [22]:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N} x[k]h[n-k] \tag{2.2}$$

where $x[n]$ and $y[n]$ represent the input and output signals to the filter. In an IIR filter, the output not only depends on current input, but also relies on previous inputs and outputs. The typical form of an IIR filter is therefore [22]:

$$\sum_{m=0}^{M} y[m]a[n-m] = \sum_{k=0}^{N} x[k]b[n-k] \tag{2.3}$$

An IIR filter is recursive in any case. While a recursive filter may have a finite impulse response, the impulse response of a non-recursive filter will always be finite.

In the frequency domain, one of the fundamental concepts to understand with a digital filter is its frequency response which represents the magnitude scaling and phase shift experienced by an input signal that is passed through the filter. In general, a magnitude response is the ratio between the amplitude of an input signal at a particular frequency and the amplitude of the output at that same frequency. Similarly, a phase response is the ratio of the phase of the input signal to that of the output signal passing through the filter at a particular frequency. Together, the magnitude response and phase response of make up the frequency response, $H(e^{jw})$, which can be characterized by the formula below [22]:

$$H(e^{j\omega}) = DTFT(h[n]) = \sum_{n=0}^{N} h[n]e^{-j\omega n} \tag{2.4}$$

So that

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \tag{2.5}$$

Note that DTFT means discrete-time Fourier transform. As shown in the above equation, the convolutive relationship between input and output in the time domain is converted into

a multiplicative relationship in the frequency domain through computing the DTFT of each signal.

The first step in designing a digital filter is selecting the key characteristics of the filter's frequency response. This essentially defines the functionality of the filter, and involves deciding which frequencies should be attenuated, boosted, etc. The target magnitude response template for a digital lowpass filter is shown in Figure 2.1. As illustrated in the figure, there are a number of key parameters of the magnitude response that must be selected, including passband corner frequency ω_p, stopband corner frequency ω_s, passband ripple δ_p and stopband attenuation δ_s. ω_p defines the area in the filter's frequency response where a signal is allowed to pass, whereas ω_s defines the beginning of the stopband region where the power of a signal is strongly attenuated. The transition band is the set of frequencies between ω_p and ω_s. Passband ripple δ_p is the amount of change in the amplitude of frequencies in the passband of the filter. Since the average passband gain is set to 1, the actual passband gain ranges between $1-\delta_p$ and $1+\delta_p$. Stopband attenuation δ_s defines the minimum attenuation level of the filter, which is usually measured with respect to the passband gain. As for phase responses, they can be classified into two main categories: linear phase and nonlinear phase. Linear phase means the phase response of the filter is a linear function of frequency within the passband, which implies that all frequency components of the input signal are delayed in time by the same amount and so the input signal is not distorted. Nonlinear phase makes the phase response a nonlinear function of frequency. Linear phase is important due to its ability to preserve the waveshape of a signal, knowing that the amplitudes of some frequencies might be changed by the action of the filter. Figure 2.2 illustrates an example of linear phase response.

2.2 QAM and OFDM Systems

As mentioned in Chapter 1, multi-carrier OFDM was introduced in DOCSIS 3.1 as a new technique that allows the throughput of DOCSIS network to be greatly enhanced. Before discussing the basic concepts of OFDM, it is mandatory to evaluate the Quadrature Amplitude Modulation (QAM) theory and review its advantages and disadvantages, since

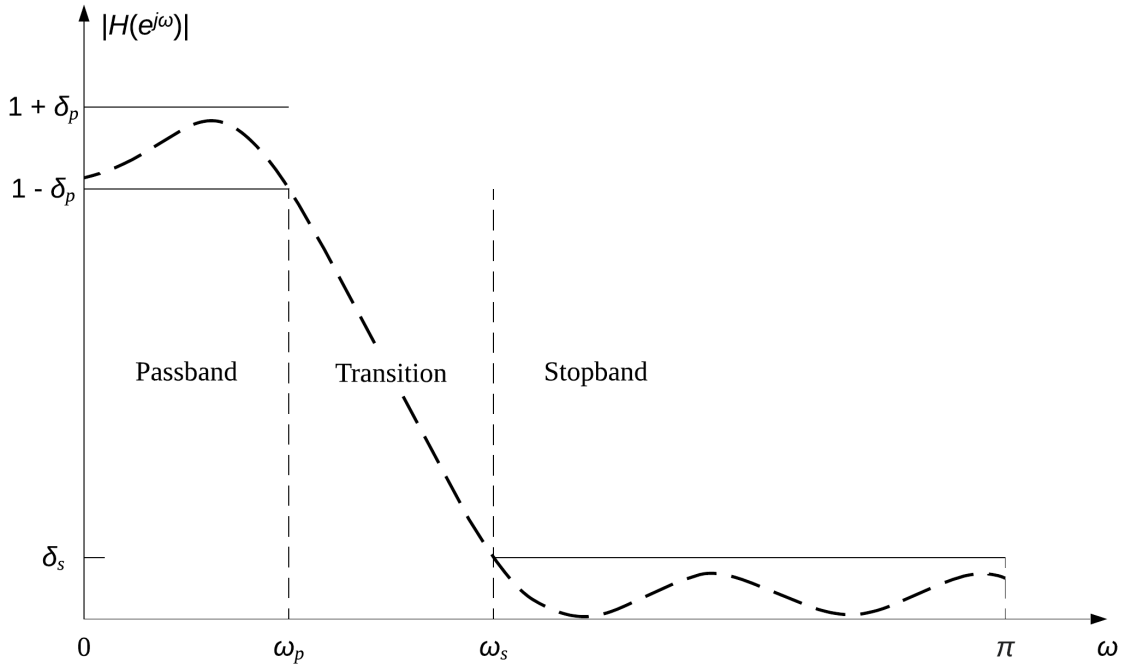


Figure 2.1: The magnitude response template for a digital lowpass filter

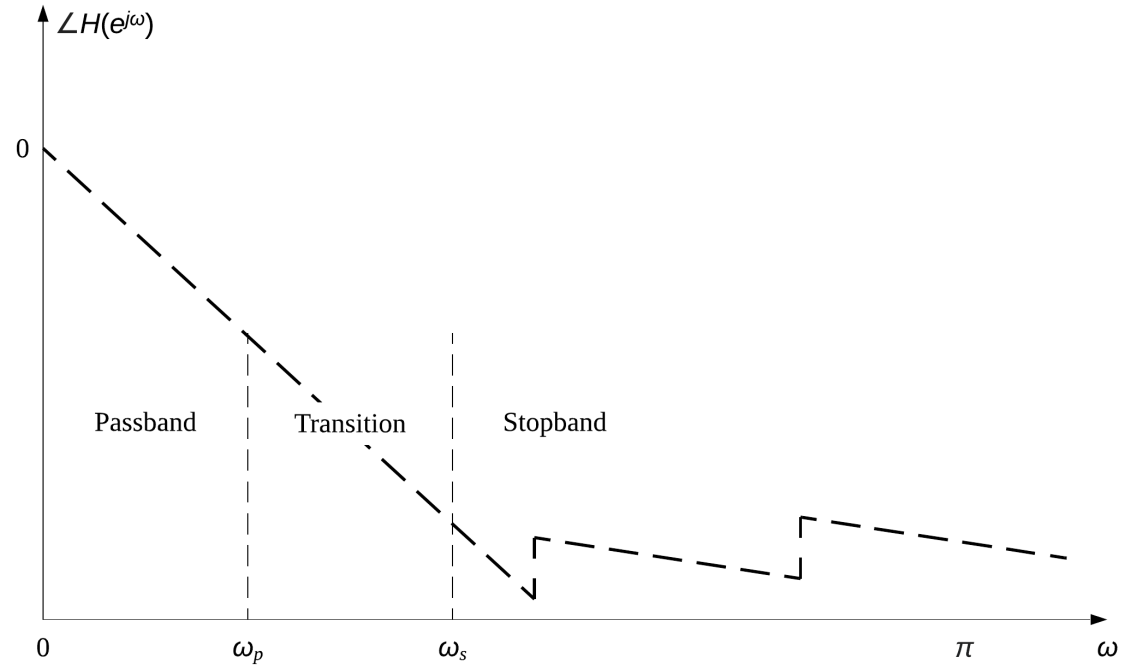


Figure 2.2: The phase response template for a linear phase digital filter

OFDM is an extension of Single-Carrier QAM (SC-QAM). SC-QAM utilizes a single carrier frequency to deliver information bits, and was widely used prior to the release of DOCSIS 3.1 specification. QAM transfers two information bit streams through varying the amplitudes of two sinusoidal carrier signals, using the amplitude-shift keying (ASK) digital modulation scheme. The two carrier signals of the same frequency f_c have a phase difference of $\pi/2$ radians, which ensures the orthogonality of the two carrier signals and avoids any interference between them. These carrier signals are usually called quadrature carriers. The sinusoids used for the modulation are typically the sine and cosine of the desired carrier frequency: $cos(2\pi f_c t)$ and $sin(2\pi f_c t)$. In addition, the scaling factors use in the amplitude modulation are commonly denoted by $V_I(t)$, which refers to the in-phase component, and $V_Q(t)$, which is the quadrature component. Adding the two scaled carrier signals results in the final transmitted signal $S(t)$ [23]:

$$S(t) = V_I(t)cos(2\pi f_c t) - V_Q(t)sin(2\pi f_c t) \tag{2.6}$$

Due to the orthogonality between the carriers, $V_I(t)$ and $V_Q(t)$ can be viewed as the real and imaginary components of a quantity known as the complex baseband signal, $S_{bb}(t) = V_I(t) + jV_Q(t)$, which represents the underlying information signal before modulation. Since $S_{bb}(t)$ is represented in the form of a complex number, its value at a given point in time can be visualized as a point on a two dimensional complex diagram where the real axis refers to the in-phase component and the imaginary axis corresponds to the quadrature component. The plot of all possible modulation symbols on a two dimensional complex diagram is usually called constellation diagram, where the symbols are known as constellation symbols, or points.

A basic block diagram of a practical QAM modulator is shown in Figure 2.3 [24]. The input binary sequence enters the QAM symbol mapper at a rate of r_b bits/s. In the QAM symbol mapper, the binary bits are allocated to multiple groups, with each group containing λ bits. These groups of binary bits are then mapped into different QAM symbols, which means that each QAM symbol sent over the channel can carry λ bits. Since there are 2^λ different combinations of λ bits, the size of the QAM symbol mapper should be $M = 2^\lambda$. A previously noted, each QAM symbol consists of an in-phase component $V_I(t)$ and a quadra-

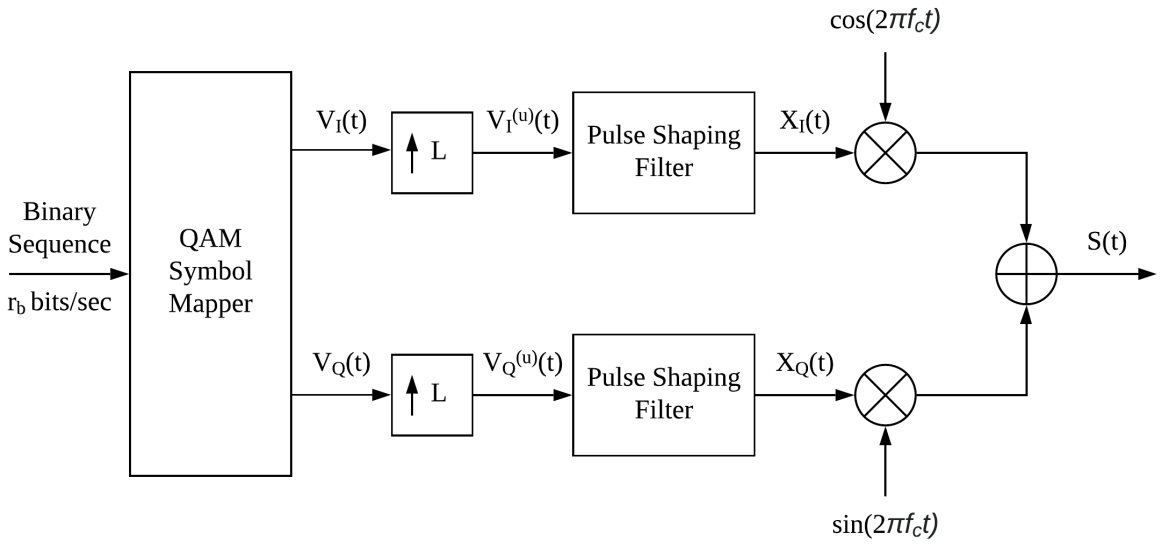
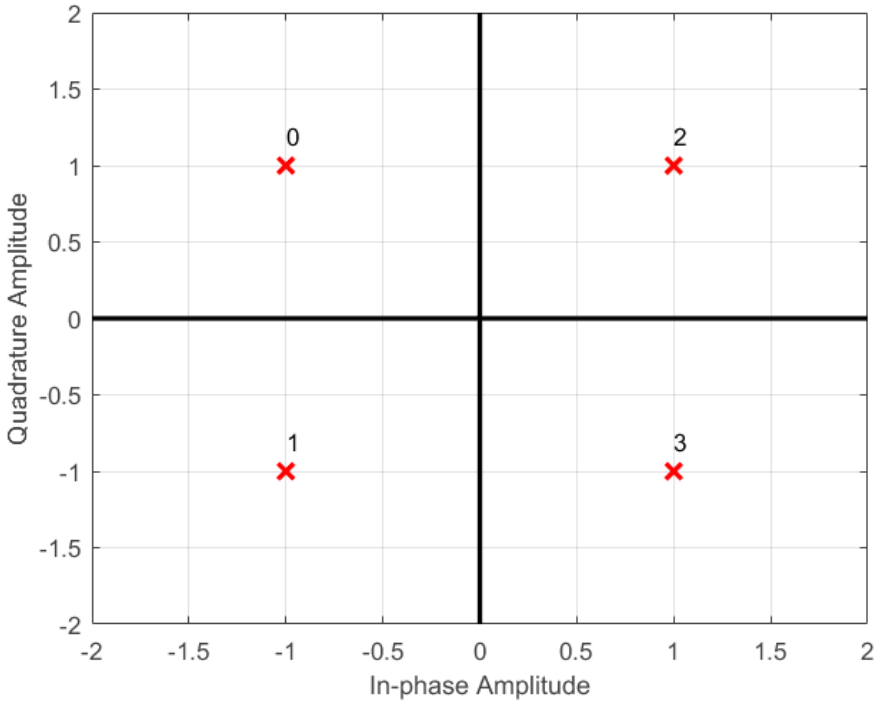


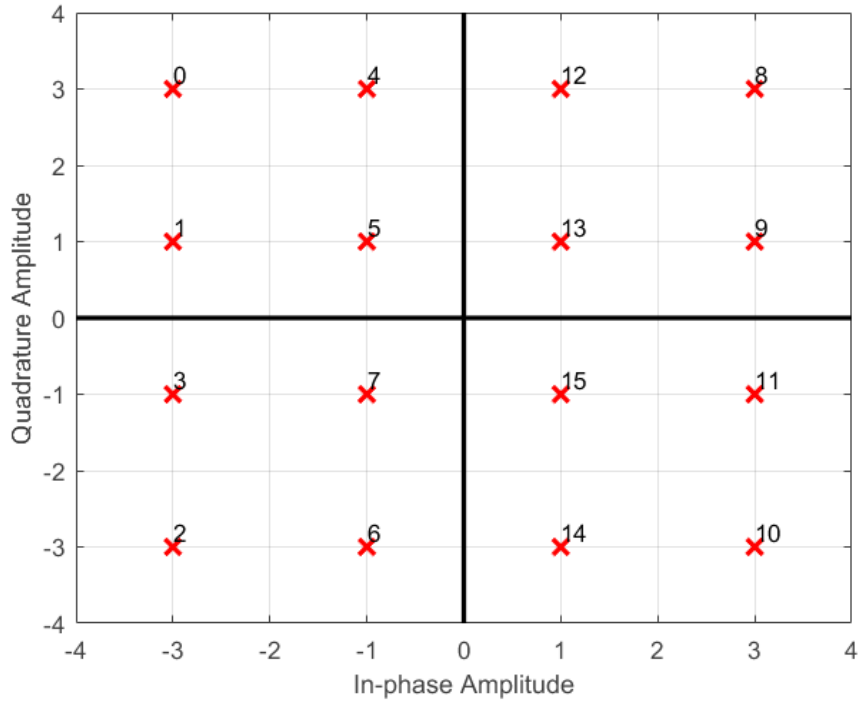
Figure 2.3: Basic block diagram of a QAM transmitter

ture component $V_Q(t)$, where t denotes the symbol index in time. Figure 2.4 demonstrates examples of constellation diagram for $\lambda = 2, 4$ [24]. The QAM symbol mapper is followed by two upsamplers that upsamples $V_I(t)$ and $V_Q(t)$ by a factor of L. Then pulse shaping filters are utilized to limit the effective bandwidth occupied by the upsampled signals $V_I^{(u)}(t)$ and $V_Q^{(u)}(t)$. Finally, the transmitted signal $S(t)$ can be found by multiplying the filtered outputs by with the sinusoidal carriers as discussed in the previous paragraph.

The structure of the QAM demodulator that corresponds to the QAM modulator in Figure 2.3 is illustrated in Figure 2.5 [24]. The received signal $R(t)$ is the sum of the transmitted signal $S(t)$ and zero-mean white Gaussian noise, which is usually denoted by $W(t)$. $R(t)$ is first multiplied by the two quadrature carriers of frequency f_c so that it can be downconverted to baseband. Then the resulting signals are processed by the matched filters that are similar to the pulse shaping filters used in the QAM modulator. The utilization of matched filters not only allows the signal-to-noise ratio (SNR) to be maximized in the presence of additive stochastic noise [24], but also eliminates the double-frequency components that are generated by multiplications with sinusoids. Following the matched filters, two downsamplers are implemented to downsample the output signals of the matched filters



(a) 4-QAM ($\lambda = 2$)



(b) 16-QAM ($\lambda = 4$)

Figure 2.4: Constellations of 4-QAM and 16-QAM

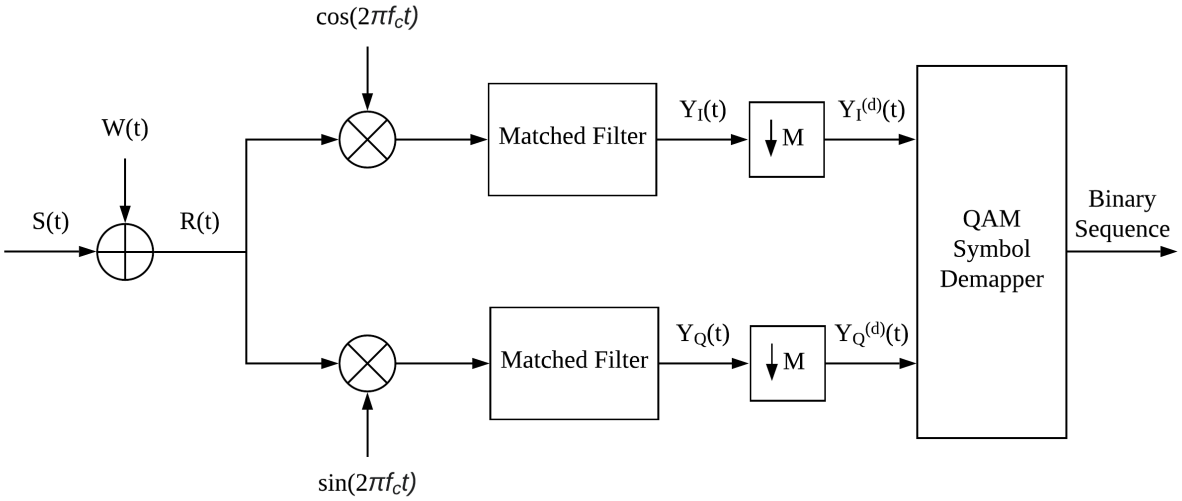
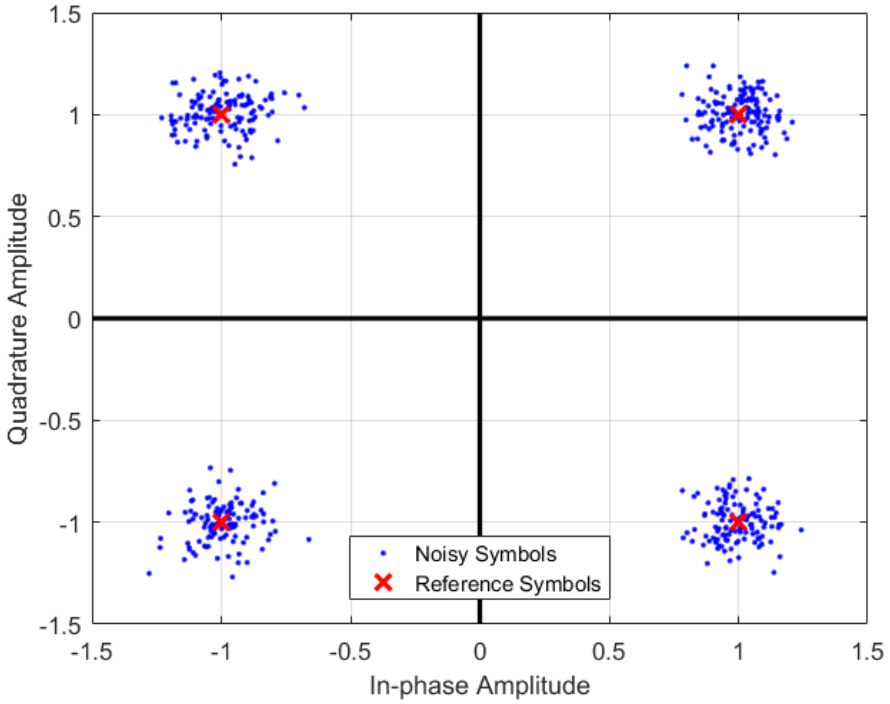


Figure 2.5: Basic block diagram of a QAM receiver

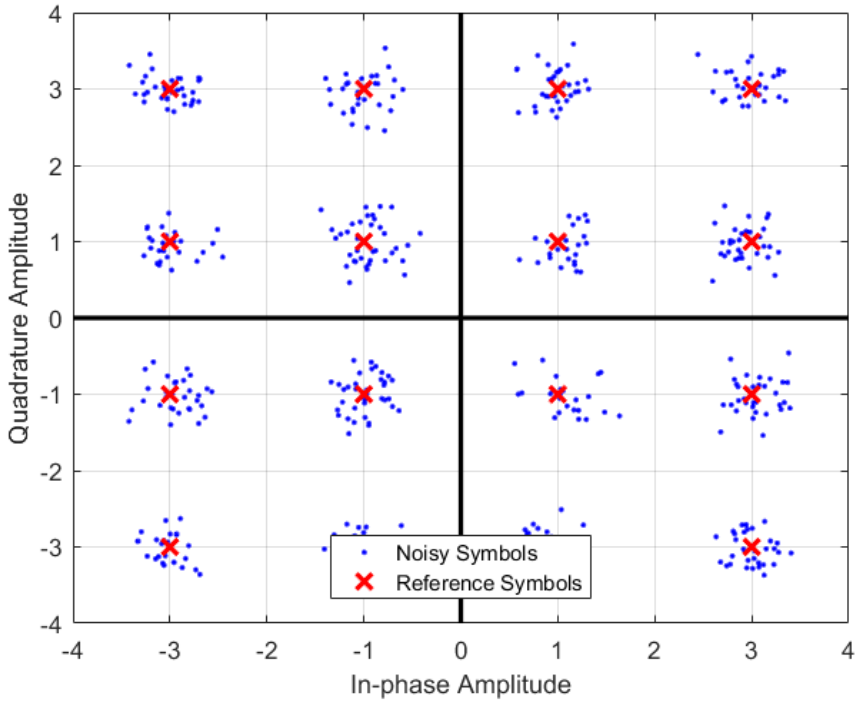
by M, resulting in the in-phase and quadrature components $Y_I^{(d)}(t)$ and $Y_Q^{(d)}(t)$. The complex signal $Y_I^{(d)}(t) + jY_Q^{(d)}(t)$ eventually enters the QAM symbol demapper, which maps each received complex symbol into the corresponding set of binary bits.

The presence of Gaussian noise has a great impact on the received constellation. Figure 2.6 shows how the received constellation symbols (blue) deviate from their ideal positions (red) after being passed through a noisy channel. This deviation makes it difficult of the the QAM demodulator to identify which symbol was transmitted and recover the original binary bits. Clearly, the performance of the receiver will depend on how how far apart the ideal signal points are and how much power is present in the noise to move the received points away from their ideal locations.

OFDM is a type of frequency-division multiplexing (FDM) scheme that is used as a digital multi-carrier modulation method. It was introduced in DOCSIS 3.1 standard to utilize the available frequency spectrum more efficiently. In OFDM, the available frequency spectrum is divided into N independent subcarrier frequencies, where the chosen subcarrier frequencies are orthogonal to each other over the duration of each transmitted symbol. Each subcarrier is then modulated with the corresponding symbol from the input data stream. The concept



(a) 4-QAM ($\lambda = 2$)



(b) 16-QAM ($\lambda = 4$)

Figure 2.6: Constellations of 4-QAM and 16-QAM with Gaussian noise

of OFDM greatly simplifies the physical structures of both the transmitter and the receiver by allowing the use of efficient Fourier transform logic.

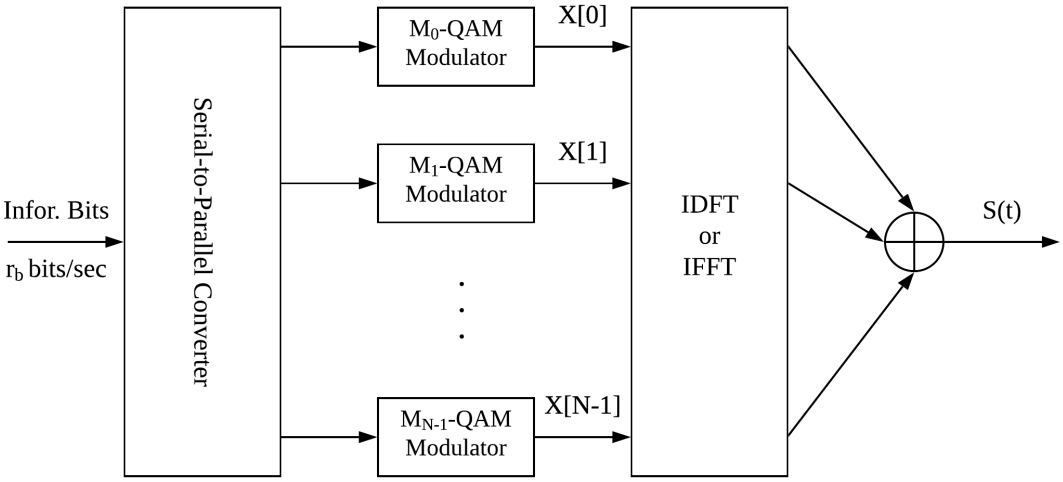
An OFDM transmitter can be built by connecting multiple QAM modulators in parallel. First, the input data stream is allocated to N groups by a serial-to-parallel converter. Each group of symbols is assigned to an independent QAM modulator. For the n^{th} QAM modulator, the n^{th} symbol in the group is modulated with the corresponding subcarrier frequency f_n. The sum of the outputs of all QAM modulators is then computed and transmitted as $S(t)$. In other words, a high-rate data stream is converted into multiple low-rate parallel data streams. An OFDM receiver can be viewed as a combination of multiple QAM demodulators. The structures of the OFDM transmitter and receiver are illustrated in Figure 2.7 [23].

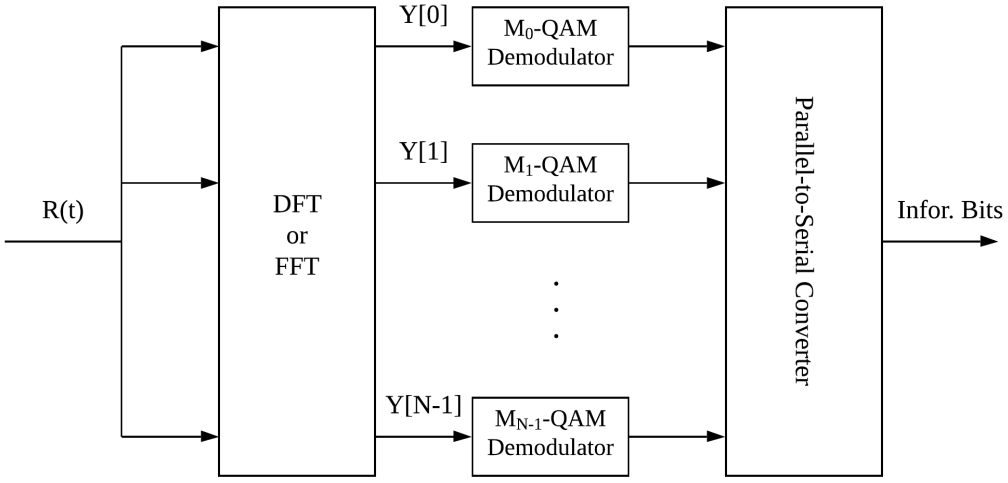
To fulfill the demand for subcarrier orthogonality, the subchannels need to be equally spaced. The minimum spacing between two adjacent subchannels is $\Delta f = 1/T_N$, where T_N is the duration of each OFDM symbol. Nevertheless, building separate QAM modulators or demodulators would require a vast number of sinusoidal oscillators and other hardware resources, which increases the implementation cost and system complexity for practical applications. An effective way to meet the demand multiple subcarrier frequencies is to utilize the discrete Fourier transform (DFT) and inverse discrete Fourier transform (IDFT). An OFDM system that is based on DFT and IDFT needs only two oscillators: one at the transmitter and one at the receiver, which greatly reduces the hardware cost and system complexity.

2.3 Filter Requirements

The main goal of this investigation is to develop a set of digital diplex filters for HFC networks. The main goal of this section is to develop and specify requirements for the filters. As discussed in the introdution, the diplexers are tightly integrated within the amplifiers which exist within HFC networks, so it is important to understand the high level design of the amplifiers in order to derive the filter requirements. Overall, the diplexer requirements may be broken down into two main categories: sampling rate requirements and signal-to-noise ratio (SNR) requirements.

(a) OFDM Transmitter





(b) OFDM Receiver

Figure 2.7: Basic block diagrams of OFDM transmitter and receiver with IDFT/DFT

2.3.1 Sampling Rate Requirements

An amplifier is defined as an electronic device that utilizes electric power from an external power supply to increase the power of a signal, allowing the signal to travel farther. The amount of power boost offered by an amplifier can be computed by its gain, which is the ratio of output voltage, current, or power to input. The gain of an amplifier is usually greater than one. Amplifiers have wide application in DOCSIS networks. They are not only used in optical nodes, as illustrated in Figure 1.4, but are also implemented in copper coaxial networks, since the signals traveling in either the upstream or downstream direction need to be amplified multiple times to reach the receivers located farthest away. Figure 1.3 describes how amplifiers are used in a coaxial network. The taps distribute a small amount of the signal power on the cable to the CMs at users' locations, while the amplifiers increase the signal power remaining on the cable to compensate for the power sent to the CMs and the power loss due to transmission over long distances.

Since the upstream and downstream signals coexist on the same copper coaxial cable, it is mandatory to use a diplexer to separate the upstream and downstream signals before amplifying them. Figure 2.8 shows the basic structure of a diplexer-based amplifier. Two directional couplers are used to separate or combine the upstream and downstream signals. As illustrated in the figure, there are two paths in the diplexer-based amplifier. The top path allows the downstream signal to pass, while the upstream signal follows the bottom path. In the top path, the ADC converts analog signal on the coaxial cable to digital signal to be sent to digital diplex filter. Then the filtered signal is sent from the digital diplex filter through a DAC before it is amplified and sent back to the coaxial cable. The operations in the bottom path are similar to those in the top path, except that all the devices are placed in the reverse order compared to the top path.

For an ADC, it is significant to define the rate at which digital values are sampled from an analog signal. This rate is commonly known as the sampling rate or sampling frequency of the converter. The digital signal that is sampled from a continuously varying band-limited signal can be sent to a DAC to reconstruct the original signal. According to the Nyquist–Shannon sampling theorem, making the sampling rate higher than twice the

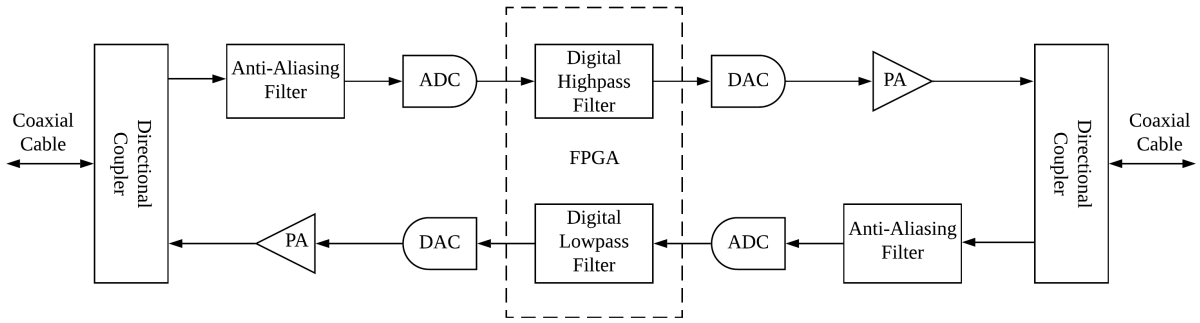


Figure 2.8: Digital-diplexer-based amplifier

highest frequency of an signal enables an accurate reconstruction of the original signal. Such sampling rate is called Nyquist rate. However, frequencies above half the Nyquist rate are incorrectly detected as frequencies below half the Nyquist rate when they are sampled. This issue is commonly referred to as aliasing. Aliasing is caused by sampling a signal at two or fewer times per cycle, which results in missed cycles and thus misdetected lower frequencies. It can be avoided by applying a lowpass filter to the input signal before sampling to remove frequencies above half the Nyquist rate. The said filter is known as an anti-aliasing filter, which is crucial for a practical ADC that is used to sample analog signals with high frequency content.

Due to the presence of aliasing, it is essential to study the frequency spectrum of the input signal to an ADC before computing the Nyquist rate. Section 1.3 states that the downstream bandwidth in a DOCSIS 4.0 network could span from 108 MHz to 1794 MHz, which means that the highest possible frequency of a downstream signal is 1794 MHz. The Nyquist rate is therefore 3588 MHz, which is twice the highest frequency. As a result, all the electronic components in the downstream path must have a sampling rate of at least 3588 MHz. Since the downstream signal occupies the top path, the passband frequency of the anti-aliasing filter in the top path must be higher than half the Nyquist rate, which is 1794 MHz. As for the bottom path, the passband frequency of the anti-aliasing filter needs to be at least 684 MHz, which is the highest frequency of the upstream bandwidth. The sampling rate, however, cannot be simply calculated by doubling the passband frequency in this case. Figure 2.9 demonstrates the frequency spectrum of a signal that has been

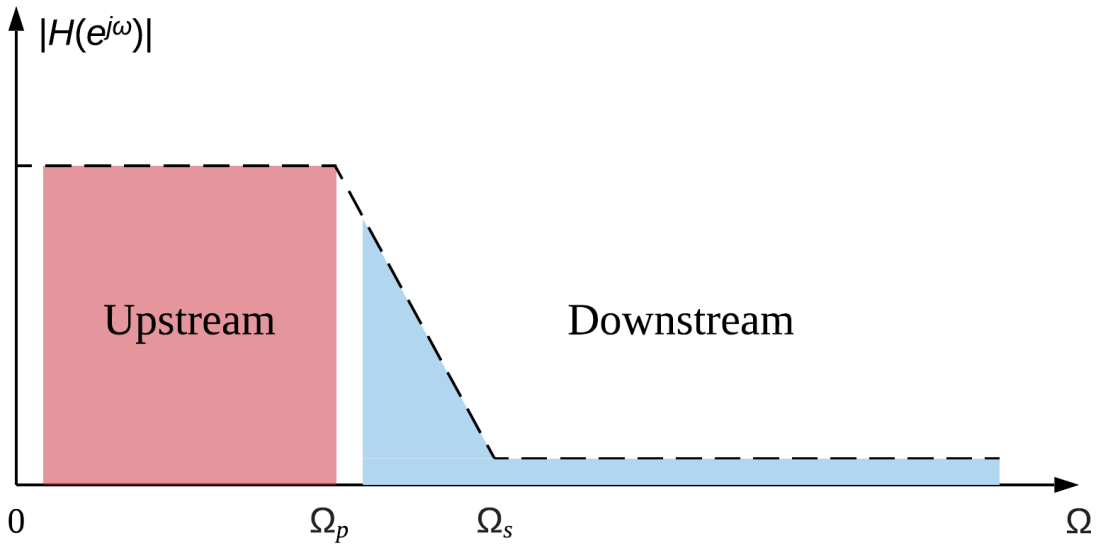


Figure 2.9: The frequency spectrum of the output signal of an anti-aliasing filter

lowpass filtered, where the dashed line represents the lowpass filter. A small portion of the downstream bandwidth remains in the filtered signal due to the presence of transition band, and could alias onto the upstream bandwidth if the sampling rate is exactly twice the passband frequency of the anti-aliasing filter. Thus, it is essential to make the sampling rate higher than twice the stopband frequency instead of the passband frequency, which ensures that aliasing does not occur. Since the stopband frequency is around 808 MHz, the sampling rate must be at least 1616 MHz. In a practical anti-aliasing filter, higher frequency content is attenuated rather than eliminated, which means that it still has effects on the sampled signal due to aliasing. This type of aliasing generates noise in the sampled signal.

2.3.2 Filter SNR Requirements

Quadrature amplitude modulation, which is also known as QAM, offers multiple important benefits for data transmission, as introduced in the previous section. Many data transmission systems, including DOCSIS 4.0 switch between different orders of QAM, such as 16-QAM, 32-QAM, etc., depending on the link conditions. Generally, higher QAM orders are more desirable since they allow for higher data rates [25]. However, the distance between

different points on the constellation diagram decreases as the QAM order rises. As discussed in section 2.2, the white Gaussian noise introduced by the channel will spread the received points on the constellation diagram, thereby decreasing the distance between received constellation points. This increases the chance for these points to be misdetected by the receiver, causing errors to occur while restoring the information bits carried by these points. In short, higher orders of QAM lead to a higher possibility of data errors being introduced [25]. Thus, it is important to find a balance between the data rate, the QAM order and the acceptable bit error rate.

To achieve reliable transmission with higher QAM orders, the received signal must have a decent SNR, otherwise data errors will occur. When the SNR deteriorates, it is necessary to either amplify the transmitted signal or reduce the QAM order to preserve the bit error rate. DOCSIS allows for many different constellations in both the upstream and downstream directions so that the QAM order can be matched to the observed SNR. The concept of making QAM order adapt to the power of white Gaussian noise is known as "adaptive modulation". Table 2.1 lists multiple DOCSIS modulation types and associated SNR threshold values [11]. Note that SNR here represents the ratio of average signal power in the occupied bandwidth to the average noise power in the same occupied bandwidth. In order for a receiver to comply with the DOCSIS standard, it must be able to achieve a specified low bit error rate when the input signal achieves the SNR levels specified in Table 2.1.

Table 2.1: DOCSIS modulation types and associated SNR threshold values

Constellation	SNR(dB)
16-QAM	15.0
64-QAM	21.0
128-QAM	24.0
256-QAM	27.0
512-QAM	30.5
1024-QAM	34.0
2048-QAM	37.0
4096-QAM	41.0

For a diplexer-based amplifier to be marketable, it should utilize the highest-order QAM that the network allows. According to Table 2.1, the highest QAM order is 4096, which corresponds to an SNR of 41.0 dB. Based on this information, it is possible to compute the passband ripple and stopband attenuation of a filter. The following analysis will reveal that SNR is dependent upon these two filter parameters. The first step in the analysis finds the mean squared error (MSE) of the signal that has been filtered by a linear phase, equiripple, lowpass filter. The lowpass filter has peak ripple δ_p in the passband and a gain between $\pm\delta_s$ in the stopband. The frequency spectrum of the signal and lowpass filter is shown in Figure 2.10. The residual power in the stopband is noise, which is bounded by:

$$P_n \leq P_{signal}\delta_s^2 \tag{2.7}$$

This noise power is the only stopband contribution to the MSE, which means that:

$$MSE_{stopband} \leq P_{signal}\delta_s^2 \tag{2.8}$$

The passband contribution to the MSE is computed in a similar way:

$$MSE_{passband} < P_{signal}\delta_p^2 \tag{2.9}$$

The total MSE is therefore:

$$MSE = MSE_{passband} + MSE_{stopband} < P_{signal}\delta_p^2 + P_{signal}\delta_s^2 = P_{signal}[\delta_p^2 + \delta_s^2] \tag{2.10}$$

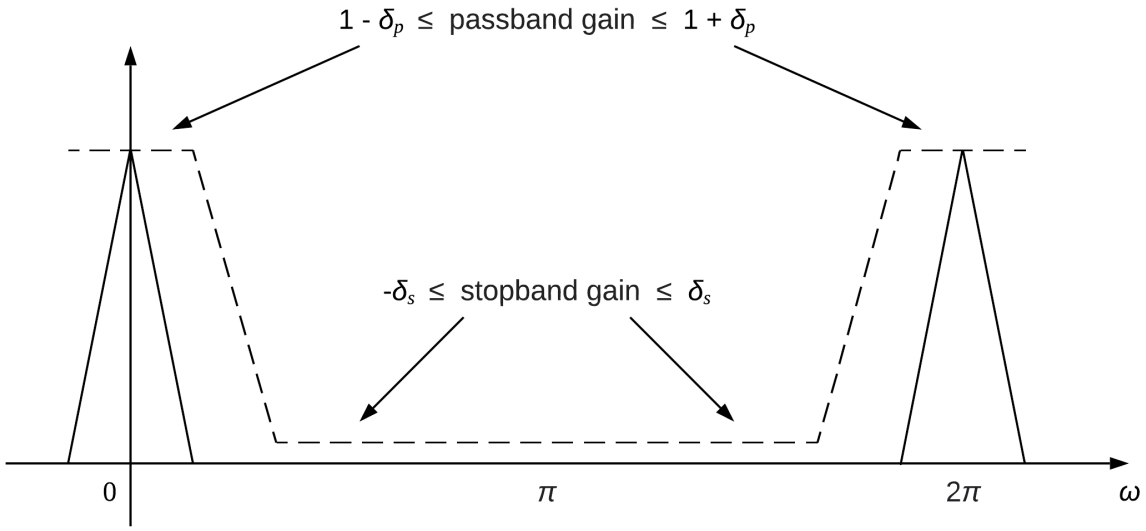


Figure 2.10: The frequency spectrum of the lowpass filtered signal

The theoretical SNR is computed below by:

$$SNR = \frac{P_{signal}}{MSE} > \frac{1}{\delta_p^2 + \delta_s^2} \tag{2.11}$$

The analysis above provides bounds on the stopband attenuation and passband ripple in order for the SNR requirement imposed by the 4096-QAM mode of DOCSIS 4.0 to be met. However, it is important to remember that the signal at the diplexer will include other sources of noise as well, including the quantization noise from the ADC, the noise and distortion from the DAC, and any noise that may be present in the input signal from the DOCSIS transmitter. When selecting specific filter parameters for an actual implementation, it is important to provide adequate margin above the target values indicated above in order to allow for such additional noise sources.

3. Filter Implementation Structures

3.1 Introduction

Filter design is the process of generating filter coefficients and structures to meet specific filtering requirements. Chapter 2 already described the filtering requirements that fulfills the demands of the digital diplex filter for DOCSIS 4.0. Once a set of coefficients has been chosen, a particular filter implementation structure is selected and used to build the system. A filter implementation structure shows how a signal propagates from the input port to the output port sample by sample. Only after both design and implementation have been performed can signals be filtered. There are many filter implementation structures that have been proposed, since the transfer function or difference equation of a filter may be realized in multiple ways. For example, evaluating a simple expression $ax + bx + c$ is equivalent to computing $(a + b)x + c$. Likewise, all filter implementation structures may be considered as different factorizations of the same transfer function or difference equation. However, the efficiency of some structures is higher since they require fewer operations or storage elements for their implementation, while others offer advantages such as enhanced numerical stability and decreased round-off error. Thus, the main objective of this chapter is to compare and contrast various filter implementation structures that meet the requirements of a digital diplex filter. Throughout the analysis, the FPGA hardware cost is used as the main basis for the comparison.

3.2 Time Domain and Frequency Domain Filtering

Periodic signals can be analyzed in two ways, or domains, which are usually called the time domain and the frequency domain. For periodic signals, time is the inverse of frequency

and vice versa. Specifically, the period of a periodic signal defines the time it takes for the signal to repeat itself, while the frequency shows how many times the signal repeats itself in a given range of time. These two variables can be used to quantify a periodic signal, and their relationship is demonstrated by the formula below:

$$\text{period} = \frac{1}{\text{frequency}} \tag{3.1}$$

Since time and frequency are the inverse of each other, time domain analysis and frequency domain analysis are inversely related in some ways.

3.2.1 Time Domain Filtering

The analysis of mathematical functions or signals with respect to time is known as a time domain analysis. In the time domain, all values in the mathematical function or signal are real numbers at different instances in the case of discrete-time or the case of continuous-time. Time domain filtering is the most straightforward way to implement a filter, where input and output are both sequences of time domain samples. There are several time domain representations for FIR filters, including the difference equation, signal flow graph and impulse response [26]. The difference equation is defined as a formula that uses past and present input samples and past output samples in the time domain to calculate present output sample. In the current application, we are interested in linear, causal, time-invariant filters, which have difference equations of the following form:

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M] \\ &= \sum_{m=0}^{M} b_m x[n-m] \end{aligned} \tag{3.2}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, and the constants b_m are called the filter coefficients. The signal flow graph is a specialized flow chart where nodes symbolize variables and branches (lines, curves, or arrows) show the functional connections between different nodes.

A straightforward approach for time domain implementation is the direct form structure, where the difference equation is evaluated directly. This form is often suitable for small

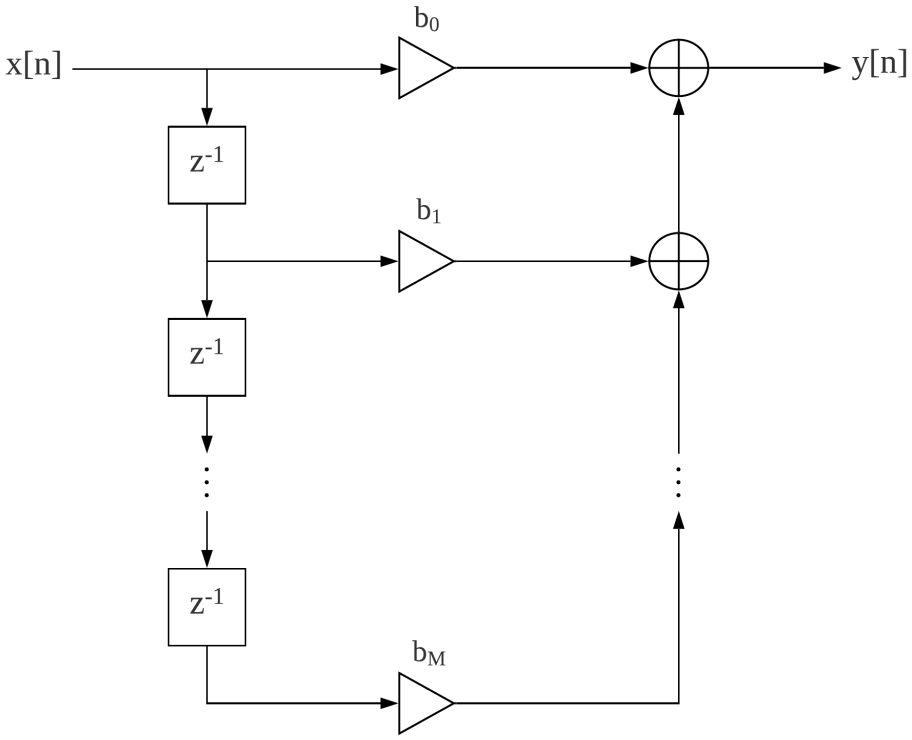


Figure 3.1: Direct form implementation of a M^{th}-order FIR filter

filters, but may be inefficient and impractical for large filters with complex designs [26]. The difference equation shown in equation 3.2 specifies the direct form implementation of a filter for the M^{th}-order case. The corresponding direct form signal flow graph is shown in Figure 3.1. Generally, the FPGA hardware cost of a direct form implementation is computed in terms of the number of required multipliers and adders. As shown in Figure 3.1, $M + 1$ multipliers and M adders are used in the direct form implementation.

For a given set of filtering requirements, the approximate number of coefficients $h[n]$ required is given by Bellanger's equation [27]:

$$M \approx \frac{2}{3} \cdot \log_{10}(\frac{1}{10\delta_p\delta_s}) \cdot \frac{2\pi}{\omega_s - \omega_p} \tag{3.3}$$

where M is the number of coefficients in the filter. In other words, the filter order is $M - 1$. Once the approximate number of coefficients is known, the specific coefficient values for a given application may be found using software routines (i.e., Matlab's FDAtool, etc) which

implement well-known algorithms such as [28] [29].

Using Bellanger's equation along with the rough filter specifications (sampling rate, passband attenuation, stopband ripple, etc) derived in Chapter 2, it can be shown that the required number of coefficients to implement the filter can be quite large. For example, a filter for the 300MHz upstream case (passband edge at 300MHz, stopband edge at 354MHz) with $\delta_s = \delta_p = 0.001$ requires 227 coefficients according to Bellanger's equation. This example shows one of the problems with the direct form structure. The hardware cost of the direct form structure increases linearly with the number of coefficients.

The other problem regarding the direct form structure is that it does not support high-speed transmission. Chapter 2 already defined the sampling rate of the input signal that enters the digital filter. Due to the properties of the direct form structure, all multipliers, adders and registers in the FPGA need to operate at the input sampling rate. Unfortunately, the maximum clock rate of a modern FPGA is approximately 400 MHz, which is much less than the input sampling rates chosen in Chapter 2. Thus, the direct form structure is not suitable for the current application.

Parallel structures can be applied to digital FIR filters to either increase the data rate of the original filter or decrease the power consumed by the original filter [30]. The formulation of parallel FIR filters requires exploiting polyphase decomposition, a scheme commonly used in multi-rate signal processing. For parallel FIR filters, the z-transform of the input signal $x[n]$ is decomposed into odd-numbered and even-numbered portions as follows:

$$\begin{aligned} X(z) &= x[0] + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} + \cdots \\ &= x[0] + x[2]z^{-2} + \cdots + z^{-1}(x[1] + x[3]z^{-2} + \cdots) \\ &= X_0(z^2) + z^{-1}X_1(z^2) \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} X_0(z) &= x[0] + x[2]z^{-1} + x[4]z^{-2} + \cdots \\ X_1(z) &= x[1] + x[3]z^{-1} + x[5]z^{-2} + \cdots \end{aligned}$$

$X_0(z^2)$ and $X_1(z^2)$ represent the z-transforms of $x[2k]$ and $x[2k+1]$ respectively, for $0 \leq k \leq$

∞. Equation 3.4 shows how $X(z)$ is decomposed into two "phases". The filter coefficients $H(z)$ can be decomposed in the same manner:

$$H(z) = H_0(z^2) + z^{-1}H_1(z^2) \tag{3.5}$$

where

$$\begin{aligned} H_0(z) &= h[0] + h[2]z^{-1} + h[4]z^{-2} + \cdots \\ H_1(z) &= h[1] + h[3]z^{-1} + h[5]z^{-2} + \cdots \end{aligned}$$

In Equation 3.5, $H_0(z^2)$ and $H_1(z^2)$ are realized as even subfilter and odd subfilter, respectively. The output signal $Y(z)$ can then be computed as:

$$\begin{aligned} Y(z) &= Y_0(z^2) + z^{-1}Y_1(z^2) \\ &= X(z)H(z) \\ &= (X_0(z^2) + z^{-1}X_1(z^2))(H_0(z^2) + z^{-1}H_1(z^2)) \\ &= X_0(z^2)H_0(z^2) + z^{-1}(X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2)) \\ &\quad + z^{-2}X_1(z^2)H_1(z^2) \end{aligned} \tag{3.6}$$

Thus, $Y_0(z^2)$ and $Y_1(z^2)$ can be written as:

$$\begin{aligned} Y_0(z^2) &= X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2) \\ Y_1(z^2) &= X_0(z^2)H_1(z^2) + X_1(z^2)H_0(z^2) \end{aligned} \tag{3.7}$$

Note that $Y_0(z^2)$ and $Y_1(z^2)$ are the z-transforms of $y[2k]$ and $y[2k+1]$ respectively. Equation 3.7 specifies the filtering operation of a 2-parallel FIR filter, which processes two input samples $x[2k]$ and $x[2k+1]$ and produces two output samples $y[2k]$ and $y[2k+1]$ every iteration. Figure 3.2 illustrates the 2-parallel FIR filtering structure. This general process can be repeated to generate arbitrary amounts of parallelism. For a K-parallel FIR filter of length M, K^2 subfiltering operations are required, each of which is of length $\frac{M}{K}$ and requires $\frac{M}{K}$ multiplications. Therefore, the K-parallel FIR filter requires $K^2 \cdot \frac{M}{K} = KM$ multipliers and $K^2 \cdot \frac{M-1}{K} = K(M-1)$ adders in total, which reveals that the hardware cost of a K-parallel FIR filter is proportional to the block size K [31]. Since a K-parallel FIR filter receives K input

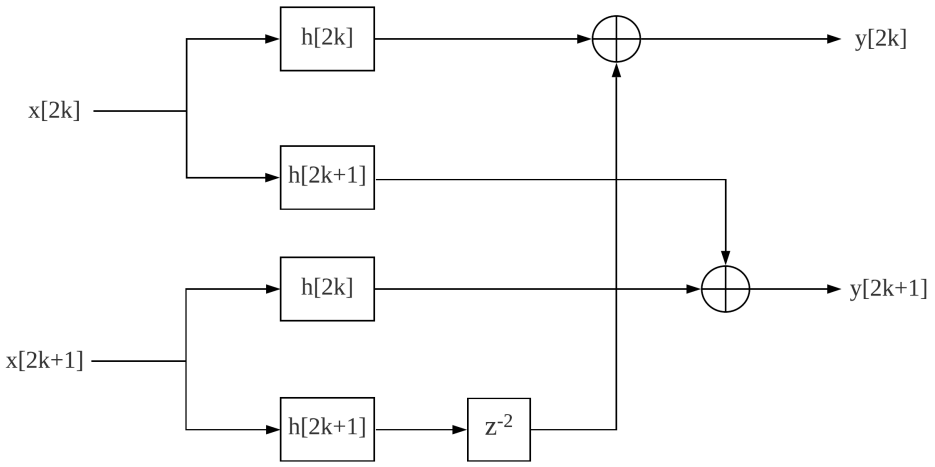


Figure 3.2: 2-parallel FIR filter implementation

samples and generates K output samples in one iteration, the input and output sampling rates of the filter are K times higher than the sampling rate of each electronic component in the filter (i.e., multipliers, adders, registers, etc). For example, assume that the electronic components in an 8-parallel FIR filter are running at the maximum clock rate of an FPGA, which is around 400 MHz. The filter is therefore able to process input samples and generate output samples at a rate of 3200 MHz, which is 8 times higher than the maximum clock rate of an FPGA. The sampling rate of a K-parallel FIR filter is also proportional to the block size K, which means that it is important to find the balance between the sampling rate and the hardware cost.

Section 2.3.1 explained that the sampling rate of the filters in the downstream direction must be at least 3588 MHz. Given that the maximum clock frequency of an FPGA is approximately 400 MHz, the minimum number of paths that are required by a parallel filter in the downstream direction is $\frac{3588}{400} \simeq 9$ in order to achieve the required sampling rate. Similarly, there must be at least $\frac{1616}{400} \simeq 4$ paths in the parallel filter implemented in the upstream direction, since the sampling rate of the filters in the upstream direction needs to be above 1616 MHz. For a 9-parallel FIR filter, there are $9^2 = 81$ subfiltering operations in total, each of which requires $\frac{M}{9}$ multiply-add operations. Hence, a 9-parallel FIR filter requires a total number of $81 \cdot \frac{M}{9} = 9M$ multipliers and $9(M-1)$ adders. The hardware cost

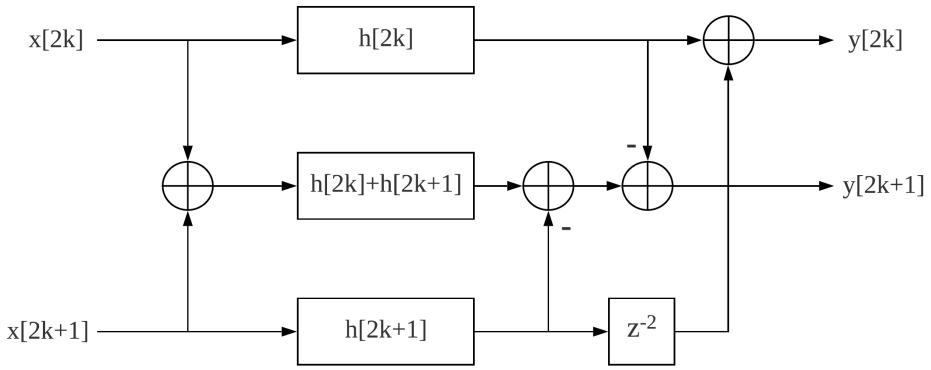


Figure 3.3: Reduced-complexity 2-parallel FIR filter implementation

of a 4-parallel FIR filter can be determined in the same manner. It requires $4M$ multipliers and $4(M-1)$ adders in total. Note that M still represents the length of filter coefficients.

The introduction of fast FIR algorithms (FFAs) allows the complexity of a parallel filter to be reduced by decreasing the number of multipliers required for implementation. This reduction in the number of multipliers comes at the cost of increasing the number of adders being used [31]. The equations below show the derivation of an FFA-based 2-parallel FIR filter:

$$Y_0(z^2) = X_0(z^2)H_0(z^2) + z^{-2}X_1(z^2)H_1(z^2) \quad (3.8)$$

$$Y_1(z^2) = [H_0(z^2) + H_1(z^2)][X_0(z^2) + X_1(z^2)] - X_0(z^2)H_0(z^2) - X_1(z^2)H_1(z^2) \quad (3.9)$$

Note that Equation 3.9 is basically Equation 3.7 written in another way. The two terms $X_0(z^2)H_0(z^2)$ and $X_1(z^2)H_1(z^2)$ exist in both Equations 3.8 and 3.9, which means that they are common terms and can be shared for the computation of $Y_0(z^2)$ and $Y_1(z^2)$. This hardware-efficient 2-parallel FIR filter is illustrated in Figure 3.3, which produces two output samples using three individual subfilters of length $\frac{M}{2}$ and 4 pre/post-processing adders in one iteration [31]. It demands $3(\frac{M}{2}) = \frac{3M}{2} = 1.5M$ multiplications and $3(\frac{M}{2} - 1) + 4 = 1.5M + 1$ additions, in contrast to $2M$ multiplications and $2(M-1)$ additions required for the traditional 2-parallel FIR filter.

Parallel FIR filters that have large block sizes can be built by cascading smaller-sized fast parallel FIR filters. According to the definition of FFA, by cascading an m-parallel FFA

and an n-parallel FFA, an $(m \cdot n)$-parallel filter can be produced [31]. The group of FIR filters that are generated by applying the m-parallel FFA are further decomposed by the application of the n-parallel FFA in a predetermined order. As a result, the group of FIR filters will be of length $\frac{M}{m \cdot n}$. One of the most important things to do while cascading the FFAs is to record both the number of multipliers and the number of adders used by the filter. The number of multipliers being used by a K-parallel FIR filter with $K = K_1 K_2 \cdots K_r$ is given by:

$$U = \frac{M}{\prod_{i=1}^{r} K_i} \prod_{i=1}^{r} U_i \tag{3.10}$$

where r is the number of required FFAs, K_i represents the block size of the FFA at iteration i, U_i is the number of filters that result from the implementation of the i-th FFA and M is the length of the filter. The computation of the number of required adders is as follows:

$$A = A_1 \prod_{i=2}^{r} K_i + \sum_{i=2}^{r} \left[A_i \left(\prod_{j=i+1}^{r} K_j \right) \left(\prod_{k=1}^{i-1} U_k \right) \right] + \left(\prod_{i=1}^{r} U_i \right) \left(\frac{M}{\prod_{i=1}^{r} K_i} - 1 \right) \tag{3.11}$$

where A_i represents the number of pre/post-processing adders used by the i-th FFA. In the previous discussion, it was revealed that the downstream path of a DOCSIS network requires 9-parallel filtering structure, which can be designed by cascading two 3-parallel FFAs. The resulting 9-parallel FIR filter would require a total of $4M$ multipliers and $4M + 54$ adders for implementation. Comparing the hardware cost of the traditional and hardware-efficient 9-parallel FIR filters, it is obvious that the hardware-efficient filtering structure offers a savings of approximately 55.56% over the traditional filtering structure. Cascading two 2-parallel FFAs would result in a hardware-efficient 4-parallel FIR filter, which is demanded by the upstream path of a DOCSIS network. A total of $\frac{9M}{4}$ multipliers and $\frac{9M}{4} + 11$ adders are required for the implementation of the hardware-efficient 4-parallel FIR filter, which represents a hardware savings of approximately 43.75% when compared to the $4M$ multipliers and $4(M-1)$ adders demanded by the traditional 4-parallel FIR filter.

3.2.2 Frequency Domain Filtering

In a frequency domain analysis, mathematical functions or signals are evaluated with respect to frequency instead of time. In other words, while a time domain graph plots the

changes of a signal over time, a frequency domain graph shows how much of the signal falls in a given range of frequencies. Frequency domain filtering is based on the Fourier transform, which is a mathematical operation that transforms a function of time, $x(t)$, to a function of frequency, $X(\omega)$. The discrete Fourier transform below transforms a series of N complex numbers into another series of complex numbers:

$$\begin{aligned} X(\omega) &= \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}\omega n} \\ &= \sum_{n=0}^{N-1} x[n] \cdot [cos(\frac{2\pi}{N}\omega n) - j \cdot sin(\frac{2\pi}{N}\omega n)] \end{aligned} \tag{3.12}$$

whereas the inverse discrete Fourier transform is given by:

$$x[n] = \frac{1}{N} \sum_{\omega=0}^{N-1} X(\omega) \cdot e^{j\frac{2\pi}{N}\omega n} \tag{3.13}$$

The Fourier transform allows a digital filter to be implemented on the frequency domain despite the fact that both filter input and output are time domain signals.

The discrete Fourier transform of a finite-duration sequence can be efficiently computed by utilizing some special algorithms that are collectively known as fast Fourier transform (FFT) algorithms. Using the DFT (or FFT), the convolution of a sequence $x[n]$ with an FIR filter $h[n]$ by the following procedure [22]:

1. Measure the lengths of the sequence $x[n]$ (L) and the filter $h[n]$ (M), then compute the length of the convolution $(N = L + M - 1)$.

2. Pad $x[n]$ and $h[n]$ with zeros to length at least N.

3. Compute the N-point DFTs $X(\omega)$ and $H(\omega)$ of $x[n]$ and $h[n]$, respectively.

4. Compute the product $Y(\omega) = X(\omega) \cdot H(\omega)$ for $0 \leq k \leq N - 1$.

5. Compute the inverse DFT of $Y(\omega)$, which results in the sequence $y[n]$. $y[n]$ is equivalent to the N-point circular convolution of $x[n]$ with $h[n]$ $(y[n] = x[n] \otimes h[n])$.

In summary, the circular convolution is equal to the inverse discrete Fourier transform of the product of the input sequences' DFTs. Since frequency domain filtering requires linear

convolution instead of circular convolution, it is important to ensure that circular convolution has the effect of linear convolution. For a finite-duration sequence $x[n]$, linear and circular convolution are equivalent. However, when $x[n]$ becomes a very long sequence, certain methods are required to establish the equivalence between linear and circular convolution.

The overlap-add method is an efficient way to calculate the linear convolution using the DFT-based method. The concept is to compute multiple convolutions of $h[n]$ with non-overlapping segments of $x[n]$ [32]:

$$x_i[n] = \begin{cases} x[n+iL], & 1 \leq n \leq L \\ 0, & otherwise \end{cases} \tag{3.14}$$

where L represents the length of an individual segment $x_i[n]$. Then compute the short convolution of each segment of $x[n]$ with an FIR filter $h[n]$ using the DFT-based method:

$$y_i[n] = IDFT_N(DFT_N(x_i[n]) \cdot DFT_N(h[n])) \tag{3.15}$$

where DFT_N and $IDFT_N$ refer to the N-point discrete Fourier transform and its inverse. The output sequence $y[n]$ is formed by overlapping the last $M-1$ samples of $y_i[n]$ with the first $M-1$ samples of $y_{i+1}[n]$ and adding the result:

$$y[n] = \sum_{i}^{\infty} y_i[n] \tag{3.16}$$

Again, M represents the length of the FIR filter $h[n]$. Note that L is customarily chosen so that $N = L+M-1$ is an integer power of 2. The FFT algorithms are used to implement the Fourier transforms for the purpose of high efficiency. Figure 3.4 shows the overall algorithm of the overlap-add method graphically, where k is the number of data blocks in $x[n]$ or $y[n]$.

The overlap–save method is another efficient way to evaluate the linear convolution of a very long sequence with an FIR filter. The concept is to insert $M-1$ zeros at the beginning of the input sequence $x[n]$, then divide the padded input sequence into overlapping blocks of length N, where the overlapping length is $M-1$. In this thesis, the overlap-add method was chosen over the overlap-save method due to its lower computational complexity and convenience of FPGA implementation.

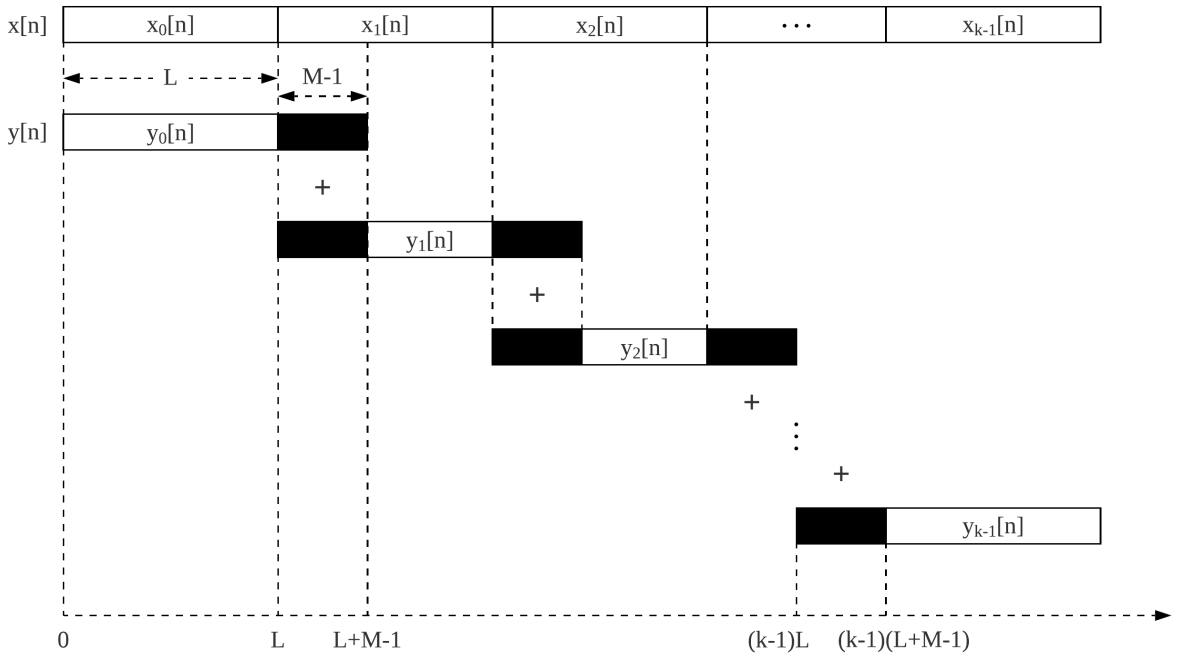


Figure 3.4: The overlap-add method: overlapping data blocks

As already mentioned above, the discrete Fourier transform of a finite-duration sequence can be computed by applying fast Fourier transform algorithms. Moreover, an FFT algorithm can be designed to have multiple parallel paths and input/output ports, which allow it to produce output samples at a rate higher than the clock rate within an FPGA. Therefore, an FFT processor can easily adapt to the high sampling rate demanded by a diplex filter.

The frequency domain filter that utilizes overlap-add or overlap-save method is generally called block-based frequency domain filter. Figure 3.5 demonstrates the basic structure of the block-based frequency domain filter described above. The filter coefficients $h[n]$ are usually fixed numbers, which means that the N-point DFT $H(\omega)$ of $h[n]$ can be predetermined and stored within the filter, reducing the total number of FFT operations being performed. Similar to time domain filter, the FPGA hardware cost of a frequency domain filter is computed in terms of the number of required multipliers and adders. As specified by the frequency domain filtering procedure above, the product of the N-point DFTs $X(\omega)$ and $H(\omega)$ of the input sequence $x[n]$ and the filter $h[n]$ is computed, which means that N is the

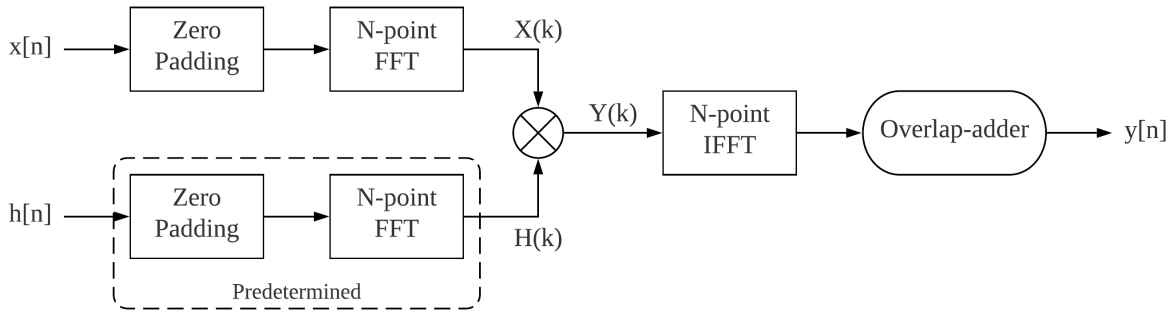


Figure 3.5: Frequency domain filter implementation

total number of multiplications that occur outside the FFT processors. The implementation of the overlap-add method requires the use of adders. As shown in Figure 3.4, the last $M-1$ samples of $y_i[n]$ are added to the first $M-1$ samples of $y_{i+1}[n]$, which reveals that the total number of adders being used outside the FFT processors is equal to $M-1$. As for the FFT processors, the amount of computation is measured in terms of the number of multiply-add operations. Theoretically, the number of multiply-add operations in an FFT processor is proportional to $N\log_2 N$ if N is an integer power of 2 [22]. Note that the multiply operations in an FFT processor refer to complex multiplications, each of which is equivalent to four real multiplications. This means that a total of $4N\log_2 N$ real multiplications are required by an FFT processor and $4N$ real multiplications occur outside the FFT processors. Likewise, the add operations in an FFT processor are complex additions, each of which corresponds to two real additions. The total number of adders used by an FFT processor is therefore $2N\log_2 N$ and $2(M-1)$ adders are required outside the FFT processors. Considering that only two FFT processors are left in the filter after predetermining $H(\omega)$, the total number of multipliers and the total number of adders required for a block-based frequency domain filter are $4N + 8N\log_2 N$ and $2M - 2 + 4N\log_2 N$, respectively. Note that this is for the N-parallel FFT case. The hardware cost of a block-based frequency domain filter may vary depending on the FFT architecture being used.

In summary, there are three filtering structures that fulfill the requirements of a diplex filter: parallel FIR filtering structure, parallel low-complexity FIR filtering structure and block-based frequency domain filtering structure. The purpose of this paragraph is to analyze

and compare these structures. Table 3.1 lists the hardware cost of each filtering structure in terms of the number of required multipliers and adders, while Table 3.2 shows the production rate of each filtering structure in samples per clock cycle. As can be observed in the tables, a parallel low-complexity FIR filter uses way less multipliers and adders than a parallel FIR filter, whereas their throughputs are exactly the same. This means that the parallel low-complexity FIR filtering structure is preferred over the parallel FIR filtering structure in this thesis. The hardware cost and throughput of a block-based frequency domain filter depends on the size of the FFT processors used in the filter. Ideally, the size of the FFT processors is set to N, where N is equal to $L + M - 1$. Tables 3.1 and 3.2 show that the decrease in the size of the FFT processors would result in the reduction in the hardware cost of the filter, which comes at the cost of decreasing the filter's throughput. Thus, the block-based frequency domain filtering structure is preferred over the other two filtering structures due to its flexible hardware cost and throughput.

Table 3.1: Hardware costs of various filtering structures

Filtering Structure	Downstream Direction		Upstream Direction	
	Multipliers	Adders	Multipliers	Adders
Parallel FIR	$9M$	$9(M-1)$	$4M$	$4(M-1)$
Parallel Low-complexity FIR	$4M$	$4M+54$	$\frac{9M}{4}$	$\frac{9M}{4}+11$
Block-based Frequency Domain (4-parallel FFT architecture)	$4N+16\log_2 N-32$	$2M+16\log_2 N+14$	$4N+16\log_2 N-32$	$2M+16\log_2 N+14$
Block-based Frequency Domain (8-parallel FFT architecture)	$4N+32\log_2 N-64$	$2M-2+64\log_2 N$	$4N+32\log_2 N-64$	$2M-2+64\log_2 N$
Block-based Frequency Domain (N-parallel FFT architecture)	$4N+8N\log_2 N$	$2M-2+4N\log_2 N$	$4N+8N\log_2 N$	$2M-2+4N\log_2 N$

Table 3.2: Throughputs of various filtering structures

Filtering Structure	Throughput (samples/clock cycle)	
	Downstream Direction	Upstream Direction
Parallel FIR	9	4
Parallel Low-complexity FIR	9	4
Block-based Frequency Domain (4-parallel FFT architecture)	4	4
Block-based Frequency Domain (8-parallel FFT architecture)	8	8
Block-based Frequency Domain (N-parallel FFT architecture)	N	N

4. Amplifier Design and Testing

4.1 Introduction

The previous chapter discussed in detail the properties of multiple filter implementation structures and compared them by reviewing their hardware costs. After careful consideration, the block-based frequency domain filtering structure was selected due to its ability to adjust hardware cost and throughput. However, as will be demonstrated later, this filter implementation structure needs to be tested before being applied to the diplex filters. This chapter discusses the testing of the FPGA-based diplex filters. First, we would discuss the general testing procedure and the software packages used for the testing. Next, the individual tests and their results would be discussed in detail.

MATLAB is a programming software designed specifically for engineers and mathematicians. The MATLAB language is considered as the core of MATLAB. It is a multi-paradigm programming language that enables the most common expression of computational mathematics. By using MATLAB, one can analyze data, manipulate matrices, create user interfaces, implement algorithms, and build models. Users may use the built-in language, applications and mathematical functions to investigate multiple methods to obtain a solution. Therefore, MATLAB has a wide application in many fields, such as image processing, engineering calculation, signal processing and communication, and control design [33]. Generally, MATLAB is used for numerical computing. An additional package named Simulink allows MATLAB to perform graphical multi-domain simulation and generate model-based design for dynamic and embedded services.

For system-on-a-programmable-chip (SOPC) or FPGA design, the Intel Quartus Prime

software is a complete design environment. Quartus is able to analyze and synthesize HDL (hardware description language) designs, which allows users to compile their designs, perform timing analysis, evaluate register-transfer level diagrams, and observe how their designs react to various stimuli. Users can also configure the target device by using the built-in programmer. In addition, Quartus offers a broad portfolio of soft and hardened IP cores optimized for FPGA devices. These FPGA IP cores can be integrated into users' designs to reduce design time and maximize performance. The HDLs that are commonly used in Quartus are VHDL (VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language) and Verilog, which are utilized in the design and verification of digital and mixed-signal systems such as FPGAs and integrated circuits. In this thesis, Verilog was chosen to model the electronic systems.

ModelSim is a multi-language software that can be utilized to examine the behavior and performance of logic circuits. The hardware description languages are VHDL, Verilog and SystemC. ModelSim has powerful waveform viewing and simulation data exporting features that are helpful for debugging and validating digital hardware designs [34]. While ModelSim can be used independently, it is able to work cooperatively with Intel Quartus Prime, Xilinx ISE or Xilinx Vivado. Simulations in ModelSim require the use of the graphical user interface (GUI). They can also be run automatically using scripts. During a simulation, users are allowed to apply inputs to the target logic circuits, and to observe the outputs generated in response. In this thesis, the functional simulation of logic circuits would be performed using the graphical waveform editing capability of ModelSim. It allows users to represent and view the input signals as waveforms.

The general procedure consists of building and simulating the block-based frequency domain diplex filter. First, the input sequence $x[n]$ and the FIR filter impulse response $h[n]$ are generated in MATLAB. Then $x[n]$ and $h[n]$ are padded with zeros and fed into the digital diplex filter developed in MATLAB. Note that the DFT of $h[n]$ is computed separately so that it can be used in the ModelSim simulation later. The filter outputs that are generated in response are stored in MATLAB temporarily for the purpose of verification. Quartus allows the digital diplex filter to be built using the Verilog language. It offers

a unique application called the FFT Intel FPGA intellectual property (IP) core, which enables the FFT processors to be implemented using the GUI. Since ModelSim may work in conjunction with Quartus, the digital diplex filter can be transferred directly from Quartus to ModelSim. After receiving $x[n]$ and the DFT of $h[n]$ from MATLAB, the simulation in ModelSim is executed, after which the simulation results are collected and compared with the filter outputs stored in MATLAB. Figure 4.1 demonstrates the general testing procedure graphically.

4.2 Filter Testing Methodology

As stated in the previous section, MATLAB is utilized to generate the input sequence $x[n]$ and the FIR filter $h[n]$. The input sequence $x[n]$ can be further divided into real input sequence $x\_real[n]$ and imaginary input sequence $x\_imag[n]$. For the initial test, $x\_real[n]$ and $x\_imag[n]$ are two groups of uniformly distributed numbers in the interval (-1,1). After converting the numbers in $x\_real[n]$ and $x\_imag[n]$ to 18-bit signed decimal numbers, $x\_real[n]$ and $x\_imag[n]$ are divided into non-overlapping segments of equal length, each of which is padded with zeros using the zeros() command. Finally, $x\_pad\_real[n]$ is formed by connecting the real zero-padded segments in series and $x\_pad\_imag[n]$ is formed by cascading the imaginary zero-padded segments. Both $x\_pad\_real[n]$ and $x\_pad\_imag[n]$ are written to a data file for the purpose of simulation. MATLAB has a built-in application called the Filter Design & Analysis Tool that allows users to generate filter coefficients by entering the filter parameters into the tool. The FIR filter $h[n]$ is thus obtained from the Filter Design & Analysis Tool. After zero-padding $h[n]$, the DFT of $h[n]$ is determined using the MATLAB command fft() and written to a data file afterwards.

In order to evaluate the results of the results of the FPGA-based fixed-point filter implementation, it is necessary to have an accurate representation of what the filter output should ideally be. To generate this a reference, the convolution of $x[n]$ with $h[n]$ was computed in MATLAB in two ways. First, a direct convolution was performed in the time domain. Second a block-based frequency domain implementation was generated, including all of the zero-padding required in the overlap-add method so that the results could be easily

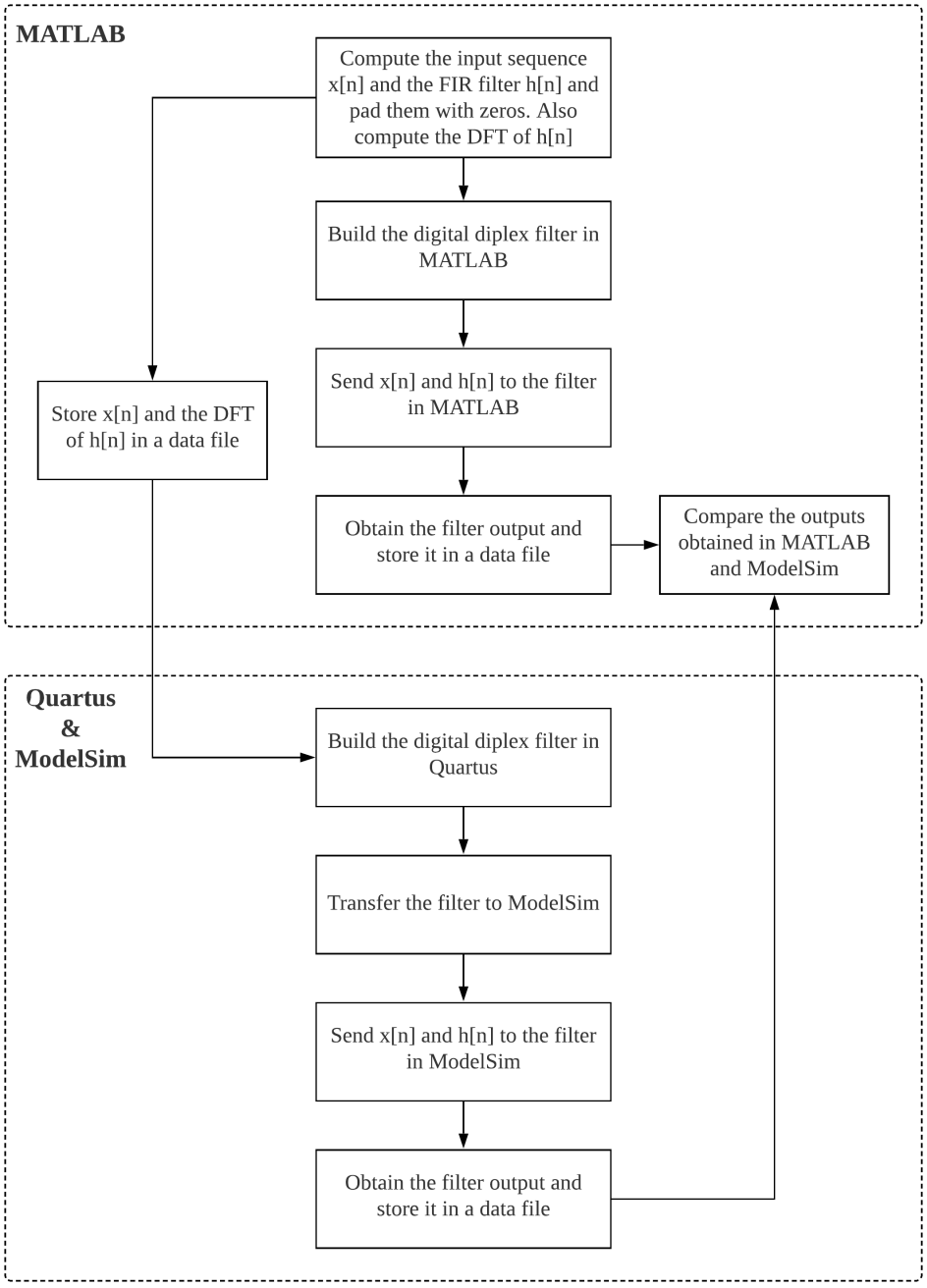


Figure 4.1: The general procedure of building and testing the digital diplex filter

compared to the corresponding results from Modelsim. Note that the signals are compared by computing the average energy of the ideal reference sequence and comparing to the average energy in the difference between the two sequences. This difference can be represented in decibels (dB), which is usually referred to as the filter implementation error ratio .

The logic circuit of the digital diplex filter is implemented using the Verilog language and compiled in Quartus. Most of the electronic components in the logic circuit, including adders and multipliers, are designed using simple Verilog commands. In contrast, the FFT and IFFT processors are built using the FFT Intel FPGA IP core. The FFT IP core is a high-performance, highly parameterizable FFT processor. It takes a complex data sequence of length N (in two's complement format) as input and generates the transform-domain complex data sequence in natural order. Table 4.1 [35] briefly describes each FFT IP core parameter, which must be determined prior to the simulations. In addition, the FFT IP core interface is shown in Figure 4.3 [35], where the input and output signals are described by Tables A.1 and A.2 [35]. The real and imaginary input data vectors of the FFT processor are $x\_pad\_real[n]$ and $x\_pad\_imag[n]$, respectively, whereas the real and imaginary output data vectors are the DFTs of $x\_pad\_real[n]$ and $x\_pad\_imag[n]$. After multiplying the DFTs of $x\_pad\_real[n]$ and $x\_pad\_imag[n]$ by the DFT of $h[n]$, the inverse DFTs of the products are computed by the IFFT processor. The inverse DFTs are then collectively stored in the random-access memory (RAM) within the FPGA. RAM is a type of computer memory that allows loading and saving in any order. The RAM interface is demonstrated in Figure 4.4. Since the data vectors stored in RAM can be read in any order, the overlap-add method can be applied to these data vectors through proper sequencing and addressing.

Table 4.2 shows the resource usage summary of the FFT processor and RAM, which is included in the compilation report offered by Quartus. Note that ALUT refers to adaptive look-up table, which is a logical construct that represents the implementation done by the combinational logic hardware of an adaptive logic module (ALM).

Figure 4.2 shows the values of some FFT IP core parameters that were used in this thesis. It also shows that the latency through the FFT IP core is 1024 samples. At an FPGA clock rate of 400 MHz, this corresponds to an absolute latency of 2.5 μs. As noted in Chapter 2,

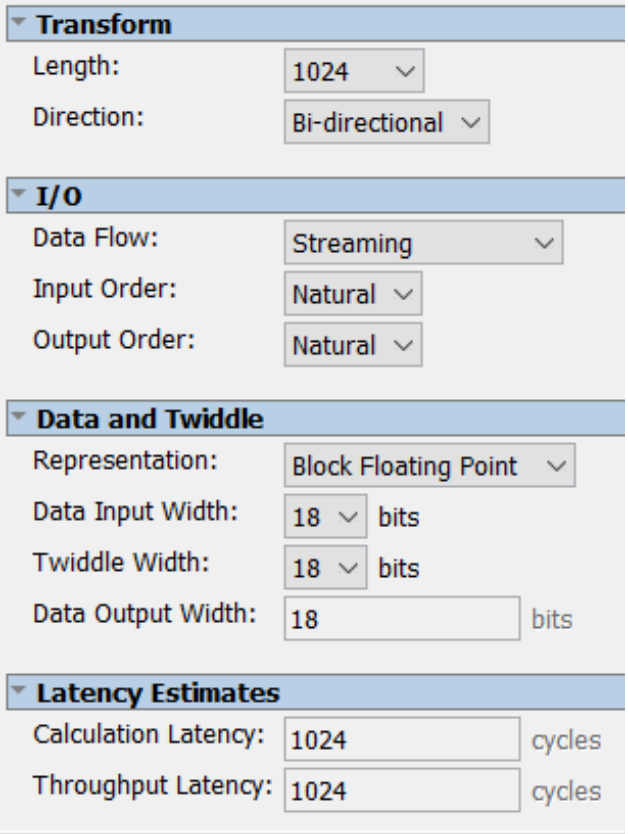


Figure 4.2: Values of some FFT IP core parameters

latency is occasionally a concern when implementing digital filters. However, the latency of the FFT IP core (2.5 μs) is much less than the target for a low-latency DOCSIS network, which is 1 ms [36]. Therefore, we can conclude that latency is not a significant issue with the FFT IP core as designed in Figure 4.2.

As already mentioned above, ModelSim can be used in conjunction with Quartus. To establish the connection between ModelSim and Quartus, a program must be run in Quartus to generate the necessary simulation libraries, ensuring an accurate model of the FPGA in ModelSim. After setting up the simulation environment, the logic circuit that is developed in Quartus can be compiled and simulated in ModelSim. The waveform viewer is utilized to observe the behaviors of signals during a simulation. Figure 4.6 shows how signals are displayed in the waveform viewer. The resulting sequence is named as $y\_modelsim[n]$ and compared with $y\_dft[n]$ in MATLAB. The comparison is also based on the difference in dB.

Table 4.1: FFT IP core parameters

Parameter	Description
Transform Length	The transform length. This value is the maximum FFT length for variable streaming.
Transform Direction	The transform direction is specifiable on a per-block basis via an input port.
I/O Data Flow	There are four types of input/output data flow: streaming, variable streaming, buffered burst and burst.
I/O Order	The input and output order for data entering and leaving the FFT.
Data Representation	The internal data representation type, only required by variable streaming FFT.
Data Width	The data precision.
Twiddle Width	The twiddle precision. Twiddle factor precision must be less than or equal to data precision.
FFT Engine Architecture	Choose between quad-output and single-output. Not available for variable streaming or streaming FFTs.
Number of Parallel FFT Engines	Choose between one, two, and four FFT engines working in parallel. Multiple parallel engines reduce transform time at the cost of device resources. Not available for variable streaming or streaming FFTs.
DSP Block Resource Optimization	Turn on for multiplier structure optimizations.
Enable Hard Floating Point Blocks	For Arria 10 devices and single-floating-point FFTs only.

Table 4.2: Resource utilization summary

Resource	Usage
Combinational ALUT	2578
Dedicated Logic Register	5735
Block Memory Bits	175760
DSP Blocks	6

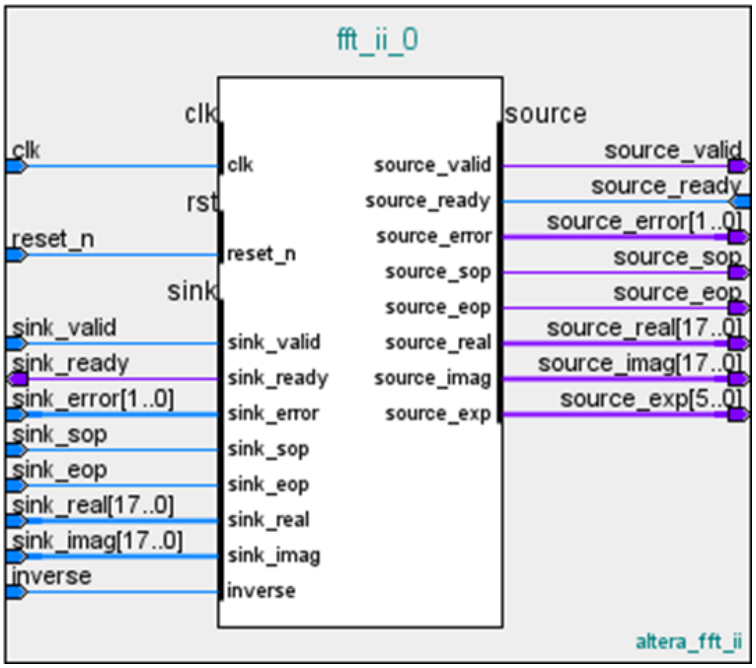


Figure 4.3: FFT IP core interface

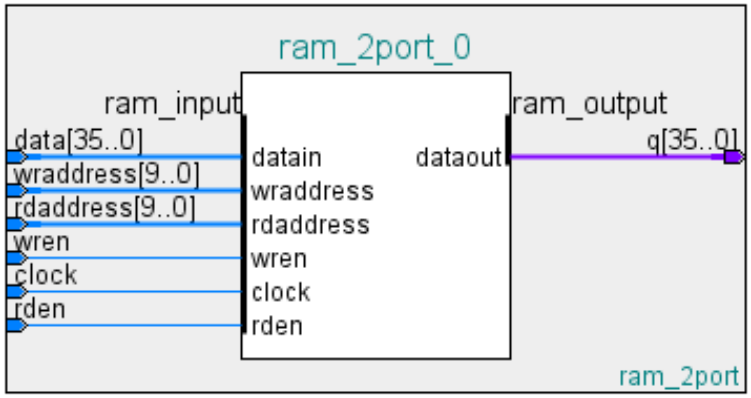


Figure 4.4: RAM interface

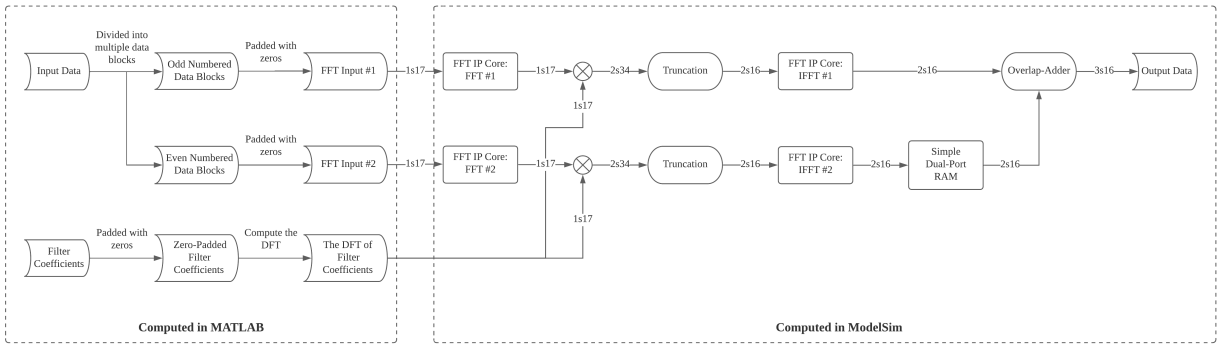


Figure 4.5: The detailed signal flow graph of the logic circuit built in ModelSim

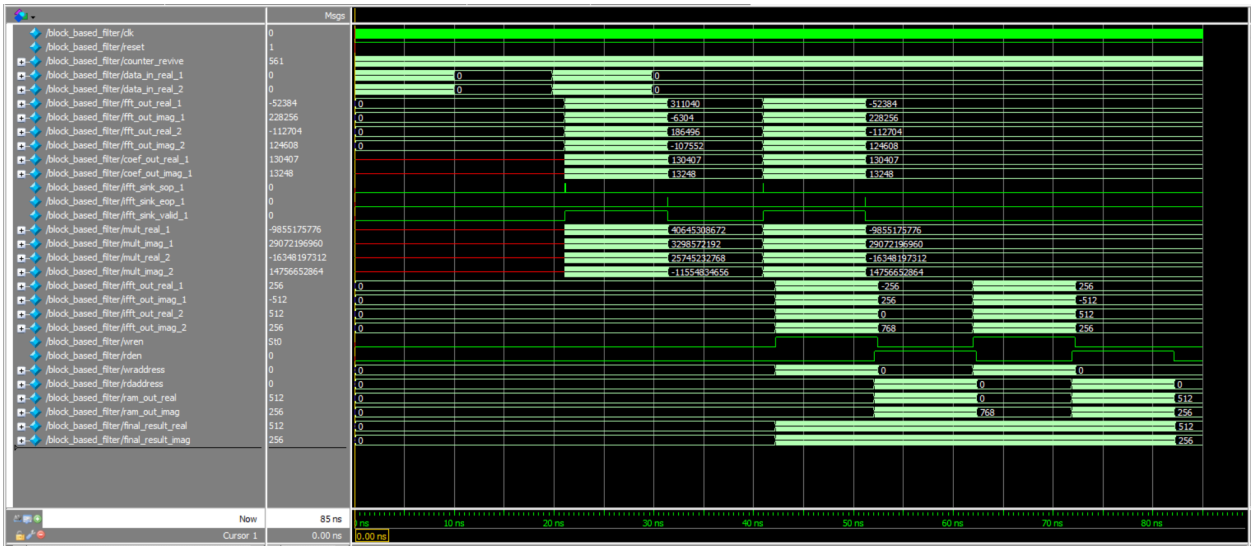


Figure 4.6: Screenshot of waveform viewer at the end of a simulation

Figure 4.5 shows the detailed signal flow graph of the logic circuit built in ModelSim. As can be seen from the figure, there are two FFT processors and two IFFT processors being used, which allows the logic circuit to process two data blocks in each iteration. The overlap-adder at the end of the circuit is thus able to produce output data blocks consistently, since it requires at least two input data blocks for each addition. The mathematical expressions on the arrows represent the formats of the numbers in the transferred signals. For example, $1s17$ means that the number format is signed decimal with 1 bit in the integer part and 17 bits in the fraction part.

Table 4.3: Frequency specifications of filters being tested

Test Number	Filter Type	Frequency Specifications	
		Passband Frequency	Stopband Frequency
1	Highpass	108 MHz	85 MHz
2	Lowpass	85 MHz	108 MHz
3	Highpass	354 MHz	300 MHz
4	Lowpass	300 MHz	354 MHz
5	Highpass	580 MHz	492 MHz
6	Lowpass	492 MHz	580 MHz
7	Highpass	808 MHz	684 MHz
8	Lowpass	684 MHz	808 MHz

4.3 Testing of Individual Filters

Figure 1.2 in the introduction of this thesis demonstrates the frequency band options available in DOCSIS 4.0. There are four options in total, each of which requires a highpass filter for the downstream path and a lowpass filter for the upstream path. Therefore, a total of eight unique filters are required to be tested. Table 4.3 shows the frequency specifications of these filters. Note that the passband ripple is fixed at 0.001 dB and the stopband attenuation was set at 80dB for all of the filters. These values were chosen to ensure a significant margin over the required SNR values specified in DOCSIS 4.0.

At the end of each simulation, two data sequences named $y\_conv[n]$ and $y\_dft[n]$ are obtained from MATLAB and written to separate text files. $y\_conv[n]$ represents the linear convolution computed by using the conv() command, while $y\_dft[n]$ refers to the linear convolution computed by using the fft() command. The difference between these two sequences can be calculated in dB using the formulas below:

$$\begin{aligned} error\ ratio &= \frac{\sum_n (y\_conv[n])^2}{\sum_n (y\_dft[n] - y\_conv[n])^2} \\ error\ ratio_{dB} &= 10 \cdot log_{10}(error\ ratio) \end{aligned}$$

The difference between $y\_conv[n]$ and $y\_dft[n]$ in each simulation can be found in Table 4.4.

Table 4.4: Analysis and comparison of simulation results

Test Number	Filter Implementation Error Ratio	
	conv() vs. fft()	MATLAB vs. ModelSim
1	109.96 dB	61.63 dB
2	100.62 dB	63.52 dB
3	109.26 dB	63.15 dB
4	106.16 dB	64.49 dB
5	108.40 dB	64.44 dB
6	108.16 dB	62.13 dB
7	107.43 dB	64.32 dB
8	109.96 dB	64.82 dB

The table shows two sets of results. The first set of results, labeled "conv() vs fft()" shows the energy in the difference between the MATLAB implementations of the time-domain filtering and the frequency domain filtering. This essentially acts as a quick check on the implementation of the overlap-add method for block-based filtering. As can be seen from the table, the difference between $y\_conv[n]$ and $y\_dft[n]$ is above 100 dB for every simulation, which reveals that $y\_conv[n]$ and $y\_dft[n]$ are almost identical and builds confidence in the chosen design based on frequency-domain filtering.

At the conclusion of the Modelsim simulation, a third data sequence that refers to the linear convolution computed within the FPGA using a fixed-point frequency domain implementation is exported from ModelSim and stored in a text file. The data sequence is named as $y\_modelsim[n]$ and compared with $y\_dft[n]$ in MATLAB. The second set of results in Table 4.4 shows the difference between $y\_dft[n]$ and $y\_modelsim[n]$ in each simulation. The average difference between $y\_dft[n]$ and $y\_modelsim[n]$ is approximately 63.56 dB. This is far better than the requirements for DOCSIS 4.0. This further proves that computing linear convolution in an FPGA using the DFT is a reliable method.

An alternative, but perhaps less scientific, way to verify $y\_conv[n]$, $y\_dft[n]$ and $y\_modelsim[n]$

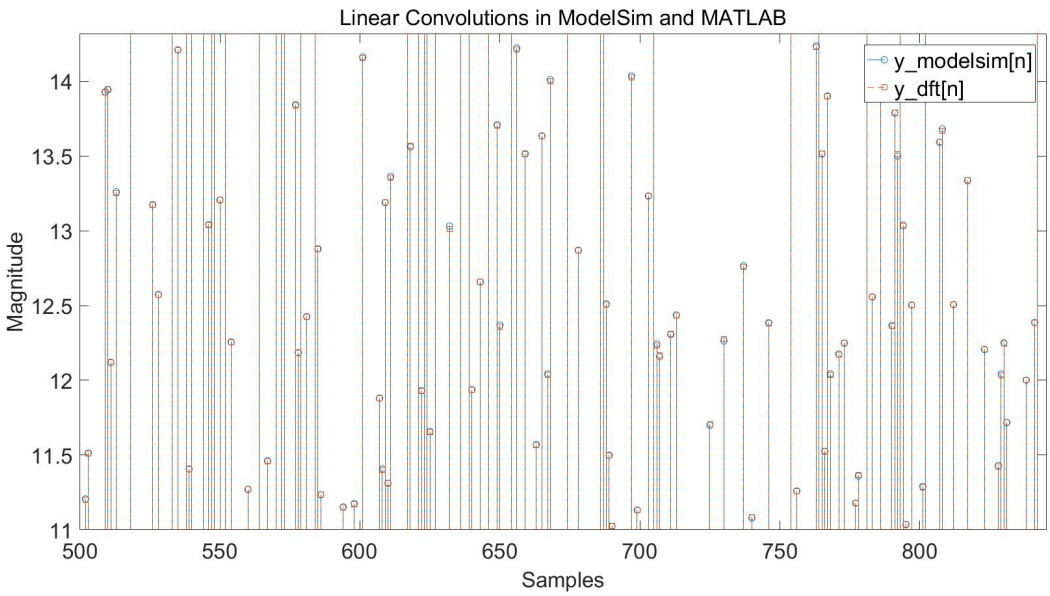


Figure 4.7: Graphical comparison between $y\_dft[n]$ and $y\_modelsim[n]$

is to compare them graphically. Figure 4.7 shows an example of graphical comparison between $y\_dft[n]$ and $y\_modelsim[n]$. A small portion of the figure has been zoomed-in appropriately for visual purposes. As demonstrated by Figure 4.7, the plots of $y\_dft[n]$ and $y\_modelsim[n]$ match each other extremely closely (so much so that it may appear there is only one set of data in the plot), thus proving that $y\_dft[n]$ and $y\_modelsim[n]$ are nearly equivalent, as desired.

The magnitude responses of the input and output signals of filters in the downstream and upstream paths are evaluated in Figure 4.8. The top figure shows the frequency content of the input signal. It is nearly constant across frequencies, which is as expected since the input signal is a random sequence. The second figure shows the magnitude response of the output signal in the downstream path, whose shape reveals that the output signal has been highpass filtered. The magnitude response of the output signal in the upstream path is shown in the bottom figure. It indicates that the output signal has been lowpass filtered. Figure 4.8 verifies that the filters designed in the previous section are functioning properly.

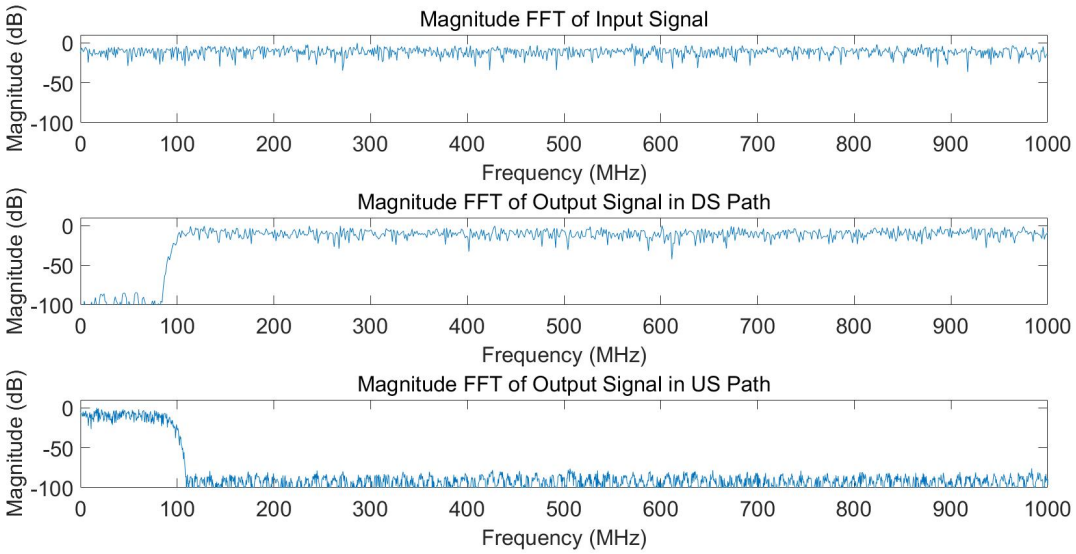


Figure 4.8: Magnitude responses of various signals

4.4 Amplifier Testing with Sinusoidal Signals

Having tested the individual filters, the main objective of this section is to build and simulate a diplexer as a whole. As indicated by Figure 2.8, a diplexer consists of two directional couplers, two amplifiers and two diplex filters. Directional couplers are electronic devices that couple a certain amount of the electric power in a transmission line to a port, allowing the signal to be utilized in another circuit. Note that directional couplers only couple the electric power flowing in one direction, which explains why a total of two directional couplers are used. In ModelSim, a directional coupler can be represented by a simple circuit that is comprised of two adders and two multipliers, as demonstrated in Figure 4.9. The amplifiers are implemented using multipliers, while the diplex filters are designed specifically for the downstream and upstream paths.

It is mandatory to verify the functionality of the diplexer before using it in practical applications. The input signals in downstream and upstream paths are first set to complex sinusoids, whose plots are easy to analyze. The frequencies of the input sinusoids are selected based on the downstream and upstream bandwidths. For the downstream path, the frequency of the input sinusoid is 800 MHz. The frequency of the input sinusoid in the

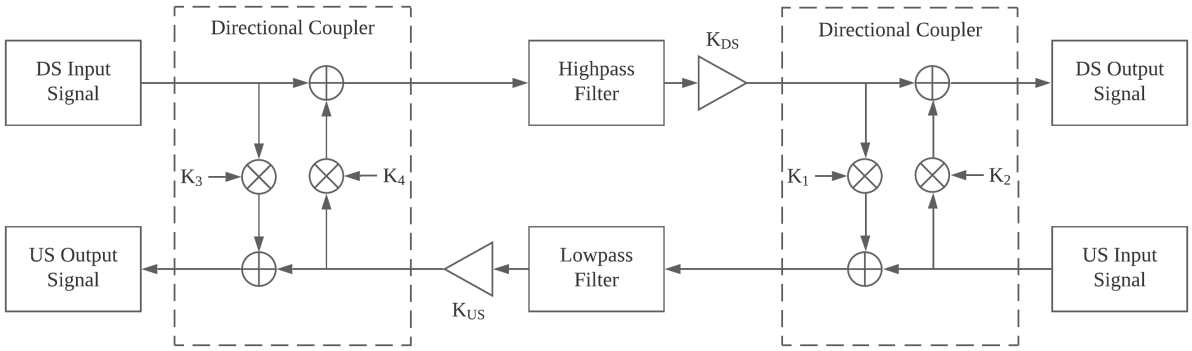


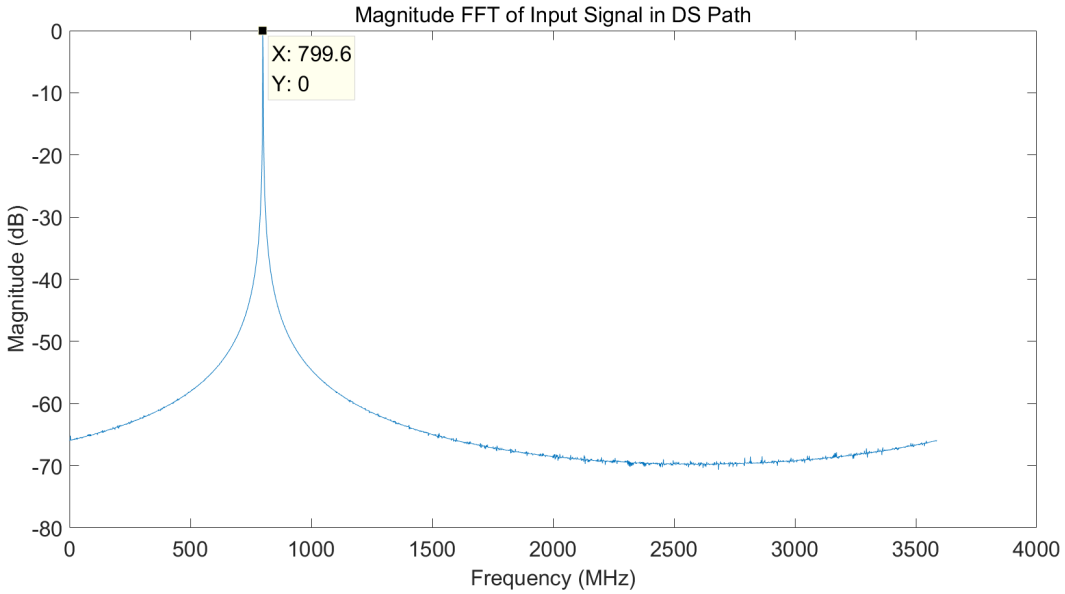
Figure 4.9: Detailed block diagram of the diplexer built in ModelSim

upstream path is 200 MHz. Note that the transition band of the diplex filters is between 492 MHz and 580 MHz in this case. Figure 4.10 shows the magnitude FFTs of the input sinusoids. Based on datasheets of comerically available directional couplers, the values of K_1, K_2, K_3 and K_4 are all set to 0.1 in order to model the signal leakage through the couplers. The diplexer is eventually simulated in ModelSim, and the output signals in downstream and upstream paths are collected and sent to MATLAB. In MATLAB, the output signals are filtered again using the predetermined filter coefficients. Then the magnitude FFTs of the filtered output signals are computed and plotted in Figure 4.11. As can be seen from the figure, the filtered output signal in downstream path has a frequency of 800 MHz, whereas the frequency of the filtered output signal in upstream path is 200 MHz. These results are as expected and lend support and credibility to the diplexer design.

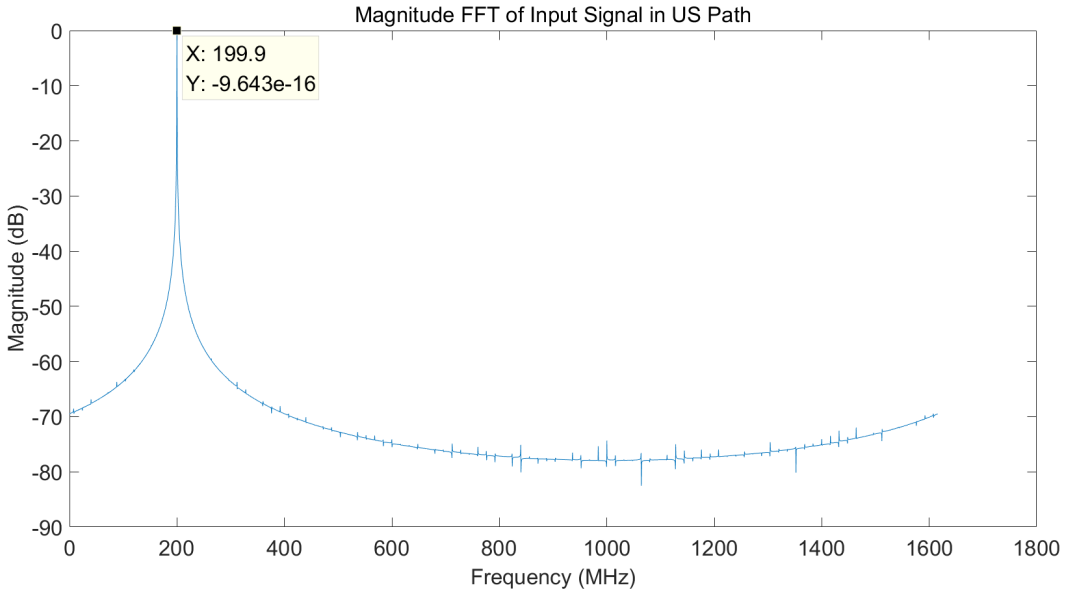
4.5 Amplifier Testing with DOCSIS 4.0 Signals

Having verified the correct operation of the digital diplexer-based amplifier using sinusoidal signals, the next step in the testing process is to confirm that the designed amplifier can successfully handle DOCSIS 4.0 signals. The requirements were described earlier in the thesis, but they are briefly restated here for clarity:

Requirement 1: The signal quality at the diplexer output (after accounting for fixed-point implementation effects such as quantization noise and directional coupler leakage)

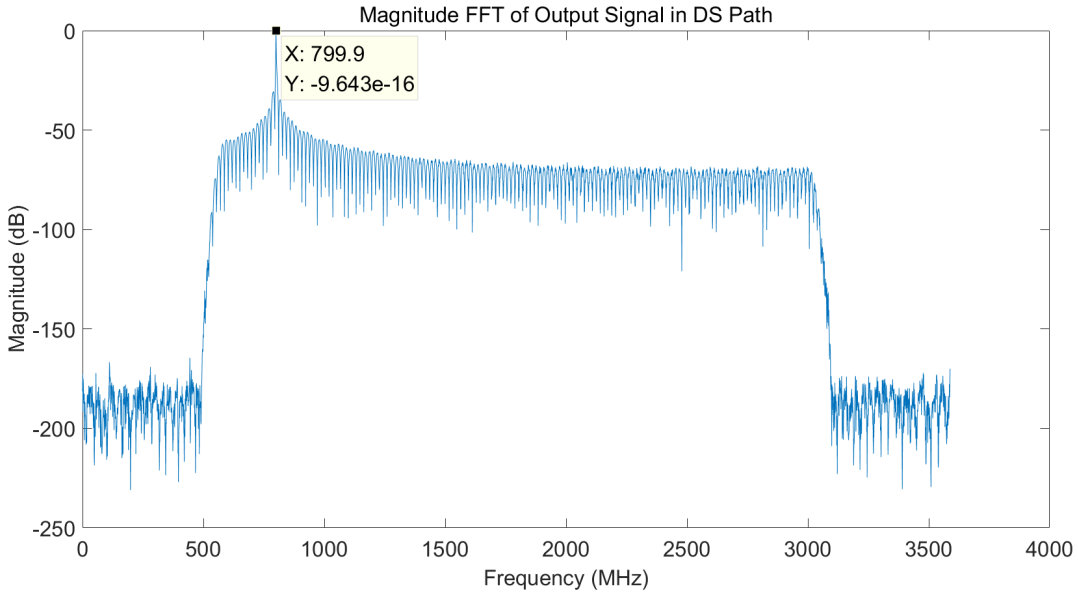


(a) Magnitude FFT of DS input signal

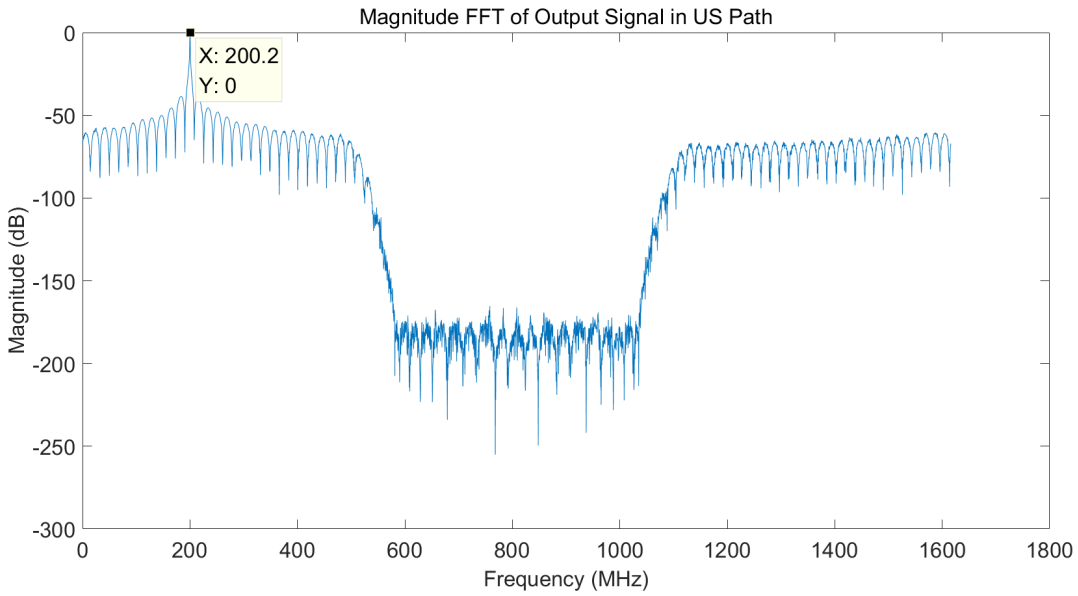


(b) Magnitude FFT of US input signal

Figure 4.10: Graphical demonstration of input signals in DS and US paths



(a) Magnitude FFT of DS output signal



(b) Magnitude FFT of US output signal

Figure 4.11: Graphical demonstration of output signals in DS and US paths

should be good enough to support the highest modulation order (and therefore highest data rate) of DOCSIS 4.0, which is 4096-QAM. As per DOCSIS 4.0, this corresponds to a Modulation Error Ratio (MER) of at least 41dB, as measured in the receiver. The details of this measurement will be discussed shortly.

Requirement 2: The diplexer's upstream/downstream transition point should be dynamically programmable to support a time-varying allocation of bandwidth to the two directions.

Requirement 3: The diplexer's design should allow for a narrower transition band between the upstream and downstream bands, as compared to a conventional diplexer built using analog components.

The following tests aim to verify that the proposed design achieves the above requirements. The general operation of each test is similar to the process described earlier in this chapter, with MATLAB being used to generate the input signals which are then processed by the designed amplifier in Modelsim. Finally, the amplifier outputs from Modelsim are analyzed in MATLAB to verify that the requirements were met.

For these tests, the input signal constructed in MATLAB is chosen to be a full spectrum of OFDM/OFDMA channels according to the DOCSIS 4.0 standard. Table 4.5 summarizes the key parameters of the DOCSIS channels used to test the amplifier. The overall input signal is created by generating a series of baseband OFDM channels, then shifting each baseband channel to a unique center frequency. The center frequencies are chosen to ensure the channels sit side-by-side and fully occupy the entire frequency spectrum.

In each test, the data from Modelsim representing the outputs from the diplexer is processed by a DOCSIS 4.0 receiver that has been constructed in MATLAB. The receiver first downconverts a selected OFDM channel from its carrier frequency down to baseband, then implements a baseband DOCSIS 4.0 demodulator. A high-level overview of the OFDM receiver is shown in Figure 4.12. After extracting an individual channel and performing timing recovery to identify the start of the OFDM symbol, the receiver performs an FFT to recover the data on the individual subcarriers. Next, the receiver estimates and compensates for any

Table 4.5: Parameters for simulated DOCSIS 4.0 channels

Parameter	US Channels	DS Channels
Subcarrier Spacing	50kHz	50kHz
FFT Length	2048	4096
Channel Bandwidth	96MHz	192MHz
Cyclic Prefix Length	192 samples	192 samples
Shaping Window Length	64 samples	64 samples

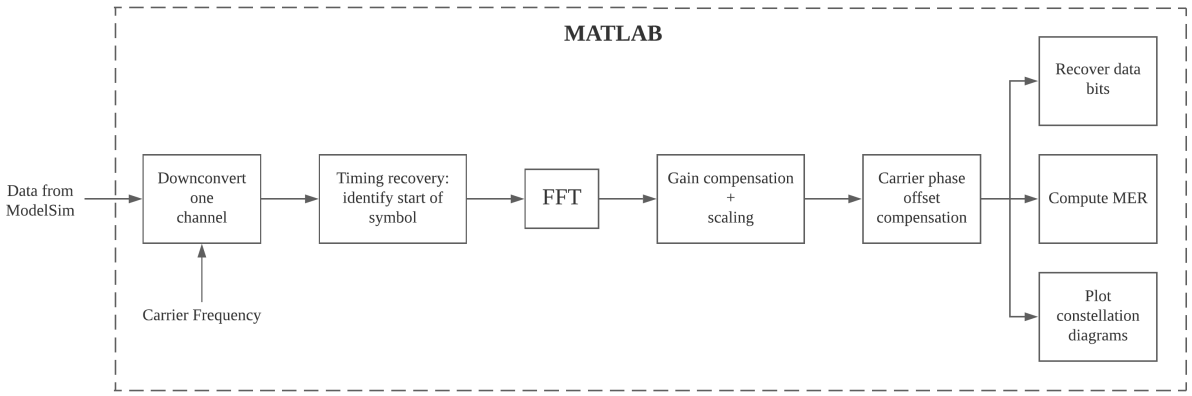


Figure 4.12: Detailed block diagram of the OFDM receiver

gain and phase differences between the received signal and the ideal received constellation. After compensating for these differences, the receiver compares the received signal samples to the expected values for an ideal 4096-QAM constellation, as quantified through the MER:

$$\text{MER} = 10 * \log_{10}\left(\frac{\sum_n(\text{Ideal constellation point for sample } n)^2}{\sum_n(\text{Error in received constellation point } n)^2}\right) \tag{4.1}$$

If the receiver constructed in MATLAB is able to successfully recover the original data without bit errors and measure the MER above the 41dB threshold, one may conclude that the signal quality through the amplifier is sufficient for high-performance DOCSIS 4.0 operation.

4.5.1 Test 1 - Basic Functionality and Dynamic US/DS Split Tests

A series of tests was conducted to verify the operation of the amplifier for three of the different upstream-downstream frequency splits outlined in DOCSIS 4.0. In particular, the following splits were tested:

- 5MHz - 300MHz upstream, 354MHz - 1700MHz downstream
- 5MHz - 492MHz upstream, 580MHz - 1700MHz downstream
- 5MHz - 684MHz upstream, 808MHz - 1700MHz downstream

Figure 4.13 shows an example of the frequency spectra of the upstream and downstream signals that were generated in MATLAB and passed into the amplifier for the 492MHz upstream case. The corresponding upstream and downstream output spectra from the amplifier device are shown in Figure 4.14. Comparing Figures 4.13 and 4.14, the leakage through the directional couplers within the amplifier can be observed. Although signals do leak between the upstream and downstream paths, the diplexer filters act upon and attenuate that leakage. Thus, the leakage appearing at the amplifier output is significantly (about 60dB) below the signals of interest, as seen in Figure 4.14. Similar results were observed for the tests with 300MHz and 684MHz upstream frequency bands. For the sake of brevity, those figures are included in Appendix B rather than the main body of this document.

As previously described, a MATLAB-based DOCSIS 4.0 receiver was used to demodulate each OFDM channel in the spectrum and compute the MER of each. The diplexer transition band (and therefore the US/DS split point) was cycled through the three options listed above and the MERs were computed in each case. The complete set of MER results is summarized in Tables 4.6 and 4.7 below. As shown in the tables, all of the MERs exceed 41dB, as desired.

Figures 4.15 and 4.16 show constellation diagrams as observed in the receiver just prior to the MER calculations. In the diagrams, the signal points corresponding to an ideal 4096-QAM constellation are plotted in black alongside the actual signal points seen by the receiver in red. It is clear from the figures that the received signal points are very close to the ideal

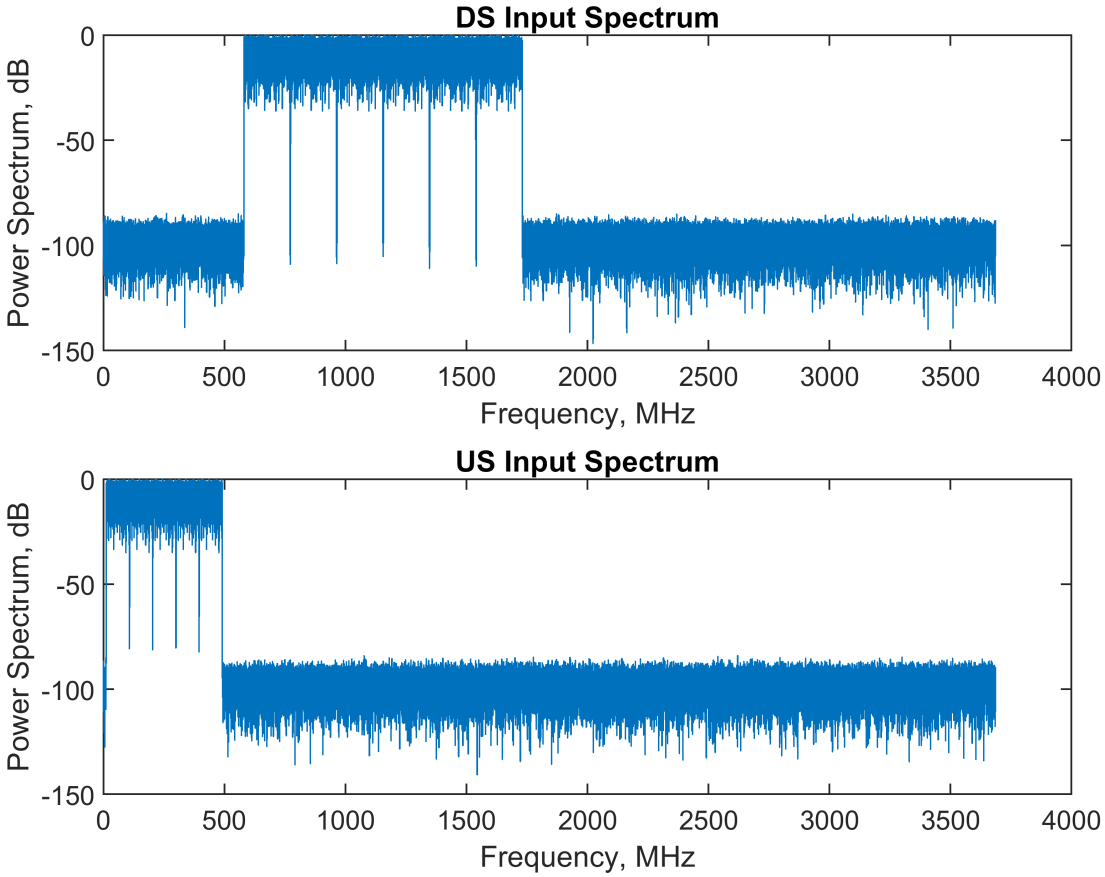


Figure 4.13: Test 1 - Upstream and downstream spectra at amplifier input (492/580 split)

Table 4.6: Test 1 - MER measurements for received downstream DOCSIS channels

300/354 Split		492/580 Split		684/808 Split	
Center Freq	MER	Center Freq	MER	Center Freq	MER
450 MHz	47.75 dB	676 MHz	47.73 dB	904 MHz	47.84 dB
642 MHz	47.81 dB	868 MHz	47.83 dB	1096 MHz	47.67 dB
834 MHz	47.75 dB	1060 MHz	47.72 dB	1288 MHz	47.83 dB
1026 MHz	47.80 dB	1252 MHz	47.84 dB	1480 MHz	47.68 dB
1218 MHz	47.74 dB	1444 MHz	47.73 dB	1672 MHz	47.85 dB
1410 MHz	47.81 dB	1636 MHz	47.83 dB	N/A	N/A
1602 MHz	47.76 dB	N/A	N/A	N/A	N/A

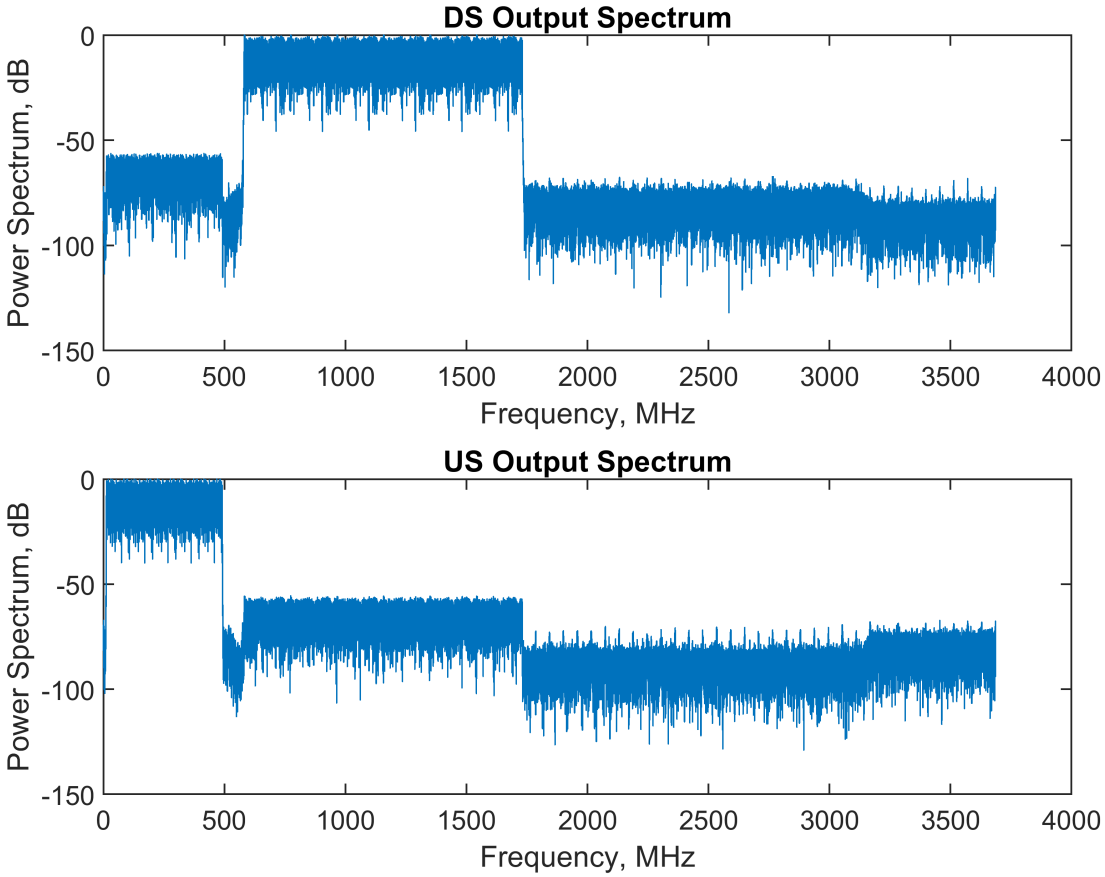


Figure 4.14: Test 1 - Upstream and downstream spectra at amplifier output (492/580 split)

Table 4.7: Test 1 - MER measurements for received upstream DOCSIS channels

300/354 Split		492/580 Split		684/808 Split	
Center Freq	MER	Center Freq	MER	Center Freq	MER
60 MHz	45.69 dB	60 MHz	45.69 dB	60 MHz	45.69 dB
156 MHz	45.70 dB	156 MHz	45.71 dB	156 MHz	45.71 dB
252 MHz	45.66 dB	252 MHz	45.67 dB	252 MHz	45.67 dB
N/A	N/A	348 MHz	45.61 dB	348 MHz	45.60 dB
N/A	N/A	444 MHz	45.69 dB	444 MHz	45.69 dB
N/A	N/A	N/A	N/A	540 MHz	45.71 dB
N/A	N/A	N/A	N/A	636 MHz	45.67 dB

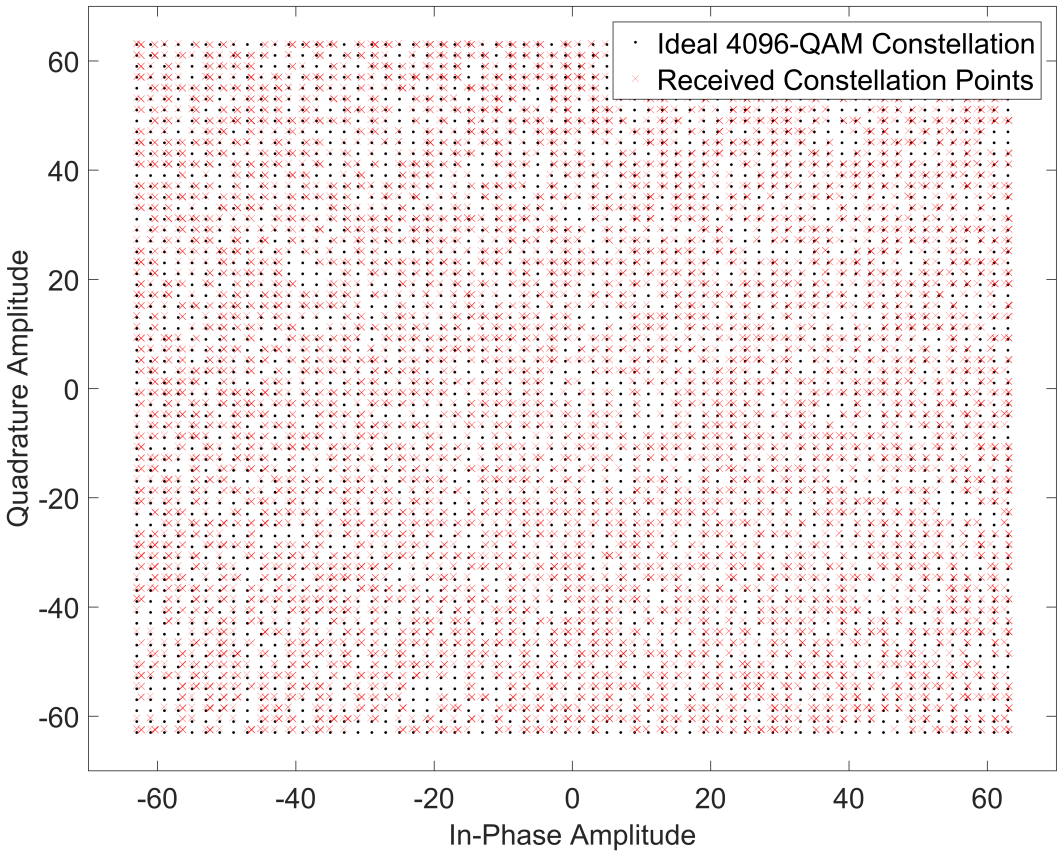


Figure 4.15: Test 1 - received signal constellation for downstream channels (492/580 split)

points for both the upstream and downstream channels. This is as expected based on the high MER values, but just provides another way to verify the correct operation of the system.

To summarize the results of these tests, we have seen that the designed amplifier can dynamically switch the frequency of the upstream / downstream split. For all of the transition options tested, the MER exceeds the 41dB target for specified in DOCSIS 4.0 for channels using 4096-QAM modulation.

4.5.2 Test 2 - Narrow Transition Band Test

A second motivation of using a diplexer based on digital filters to construct an amplifier for DOCSIS networks is to reduce amount of spectrum which must be left unoccupied to accommodate the transition bands of the filters. It is clear that digital filter hardware allows

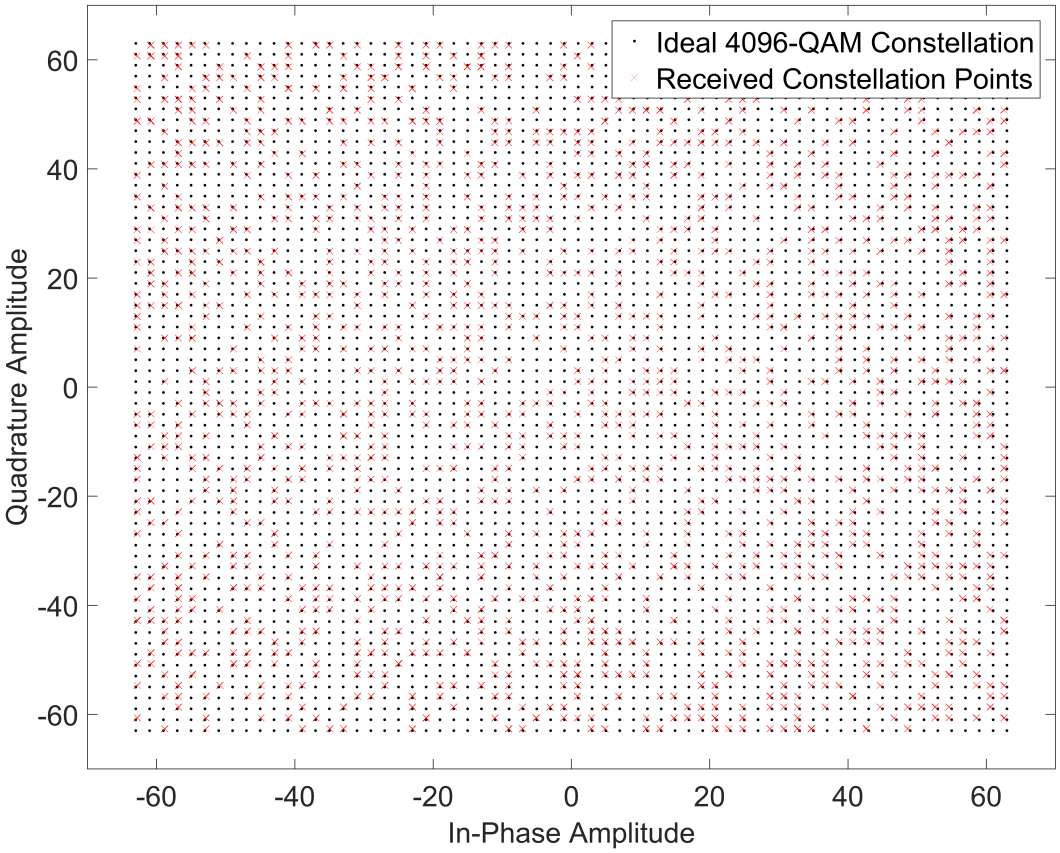


Figure 4.16: Test 1 - received signal constellation for upstream channels (492/580 split)

the construction of diplex filters with narrower transition bands (for a hardware cost) than can be achieved with analog components as in current DOCSIS equipment. However, due to imperfections in the transmitter (such as phase noise and quantization), it is not obvious whether the signal quality in the "extra" bandwidth obtained in the transition band would be good enough to support DOCSIS 4.0 transmission.

Therefore, this test aims to verify that a DOCSIS network using diplex filters with narrow transition bands will actually yield an increase in the overall network capacity. The approach taken is to construct a "real-life" model of a DOCSIS transmitter by modeling the impairments described above. In particular, a transmitter which meets the DOCSIS 4.0 standard may generate in-band noise, distortions, and spurious tones which are up to 45dB below the main desired signal (but no higher). To model the worst-case situation, which is a transmitter which just barely meets the specification, the MATLAB signal generator discussed in the previous section was modified to add distortions to each transmitted channel at a level 45dB below the main signal.

The diplexer used in this test was designed to have a much sharper transition band than that required by DOCSIS 4.0. To best demonstrate the effect, the largest possible upstream band was selected (5-684MHz). For a 684MHz upstream, DOCSIS 4.0 typically would place the first downstream channel at 808MHz (a 124MHz transition band), but the diplexer in this test was designed for a 48MHz wide transition band. The lowest downstream channel was placed at 732MHz, thus gaining 76MHz of usable spectrum as compared to the DOCSIS specifications. Figures 4.17 and 4.18 show the spectrum at the input and output of the amplifier.

As in the previous tests, Table 4.8 shows the MER observed in the receiver after each of the channels was downconverted and demodulated. Of particular interest in this test are the highest frequency upstream channel (center frequency of 636MHz, upper edge at 684MHz) and the lowest frequency downstream channel (center frequency of 828MHz, lower edge at 732MHz), as these are the channels most vulnerable to any noise or distortion that is not fully removed by the filters. Constellation diagrams of the received signals are available in Appendix B. Table 4.8 indicates that all of the channels exceed the 41dB MER target.

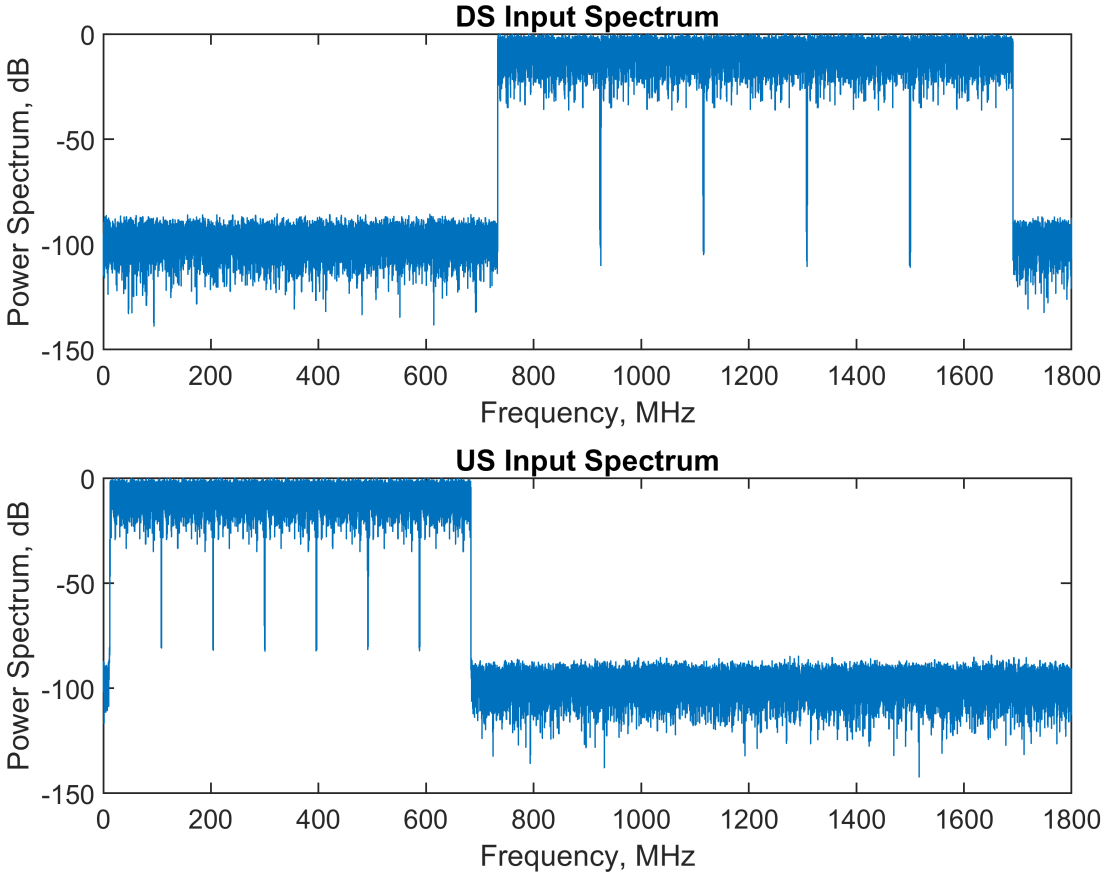


Figure 4.17: Test 2 - Upstream and downstream spectra at amplifier input (684/732 split)

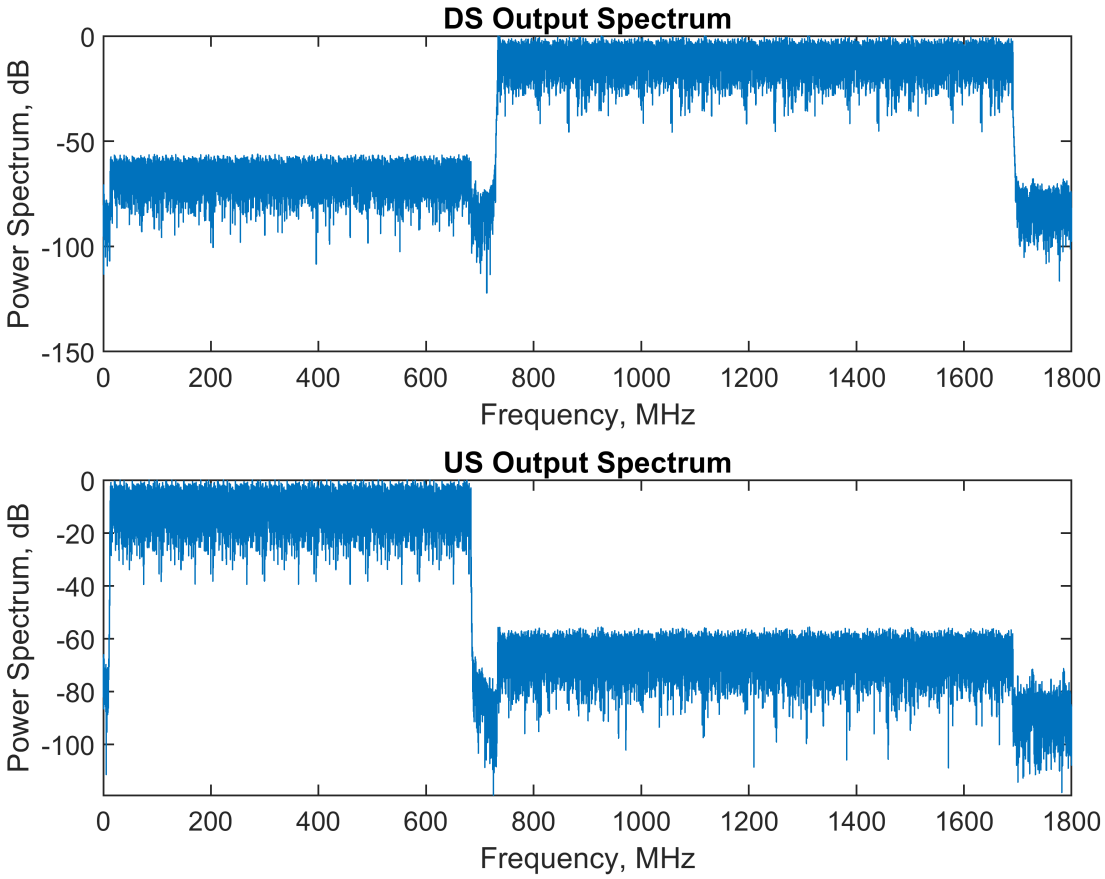


Figure 4.18: Test 2 - Upstream and downstream spectra at amplifier output (684/732 split)

Table 4.8: Test 2 - MER measurements for received channels

Upstream Channels		Downstream Channels	
Center Freq	MER	Center Freq	MER
60 MHz	42.19 dB	828 MHz	44.31 dB
156 MHz	42.21 dB	1020 MHz	44.25 dB
252 MHz	42.17 dB	1212 MHz	44.31 dB
348 MHz	42.11 dB	1404 MHz	44.26 dB
444 MHz	42.19 dB	1596 MHz	44.31 dB
540 MHz	42.21 dB	N/A	N/A
636 MHz	42.17 dB	N/A	N/A

Furthermore, there are no significant variations in performance between channels. These observations verify the concept that the use of digital diplexers can help to recover spectrum within the DOCSIS network.

4.5.3 Test 3 - Practical Transmitter Test

This final test expands upon the concept of transmitter impairments from section 4.5.2 by adding not only in-band (within channel) noise and distortions, but also out-of-band (outside of the each channel) noise and distortions. The same impairment level of 45dB was used for this test, corresponding to a worst-case DOCSIS transmitter.

For this test, a standard 492/580 upstream/downstream split as described in DOCSIS 4.0 was used. The corresponding upstream and downstream spectra at the input and output of the amplifier are shown in Figures 4.19 and 4.20. A careful comparison of Figures 4.19 and 4.20 against the corresponding figures from sections 4.5.1 and 4.5.2 illustrates the effect of the out-of-band distortions. It is clear that the wideband noise level modeled this test is much increased from that in the previous tests (which was due solely to quantization noise).

As before, the MER of each upstream and downstream channel was measured in the MATLAB-based receiver. The results are presented in Table 4.9. All of the channels are

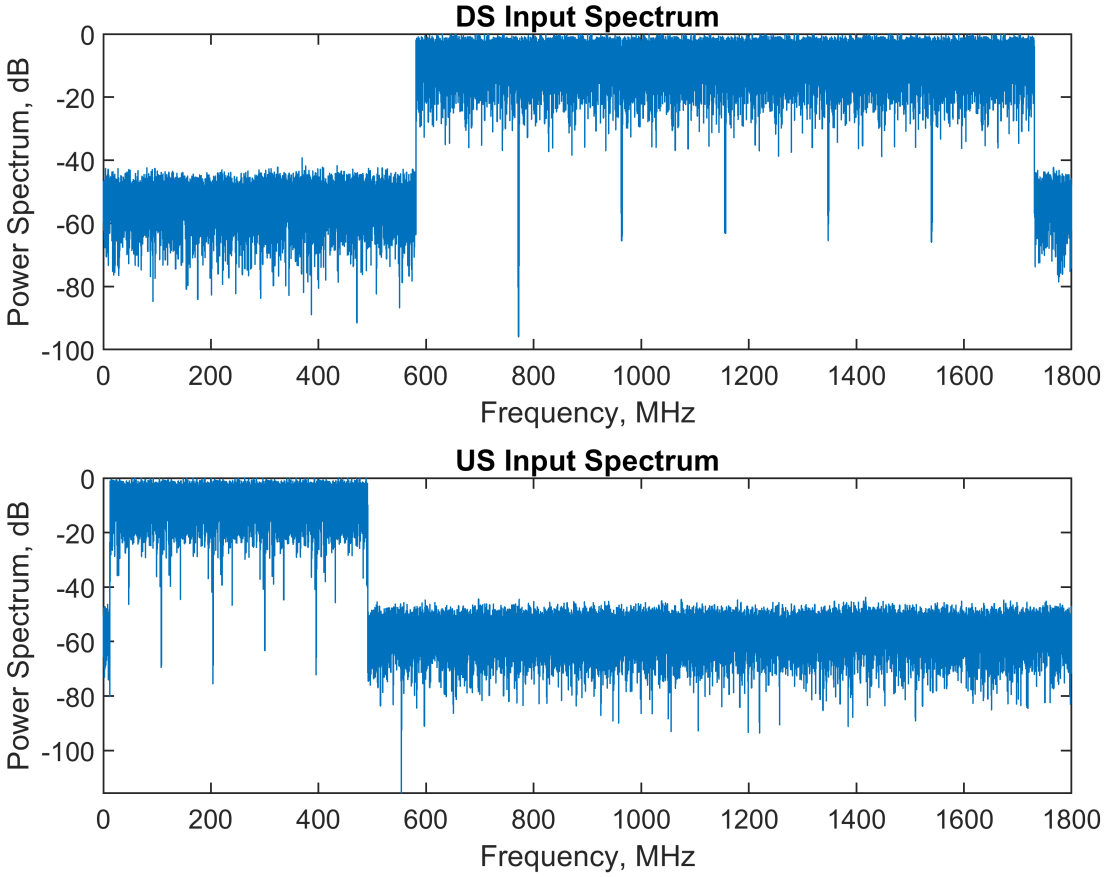


Figure 4.19: Test 3 - Upstream and downstream spectra at amplifier input (492/580 split)

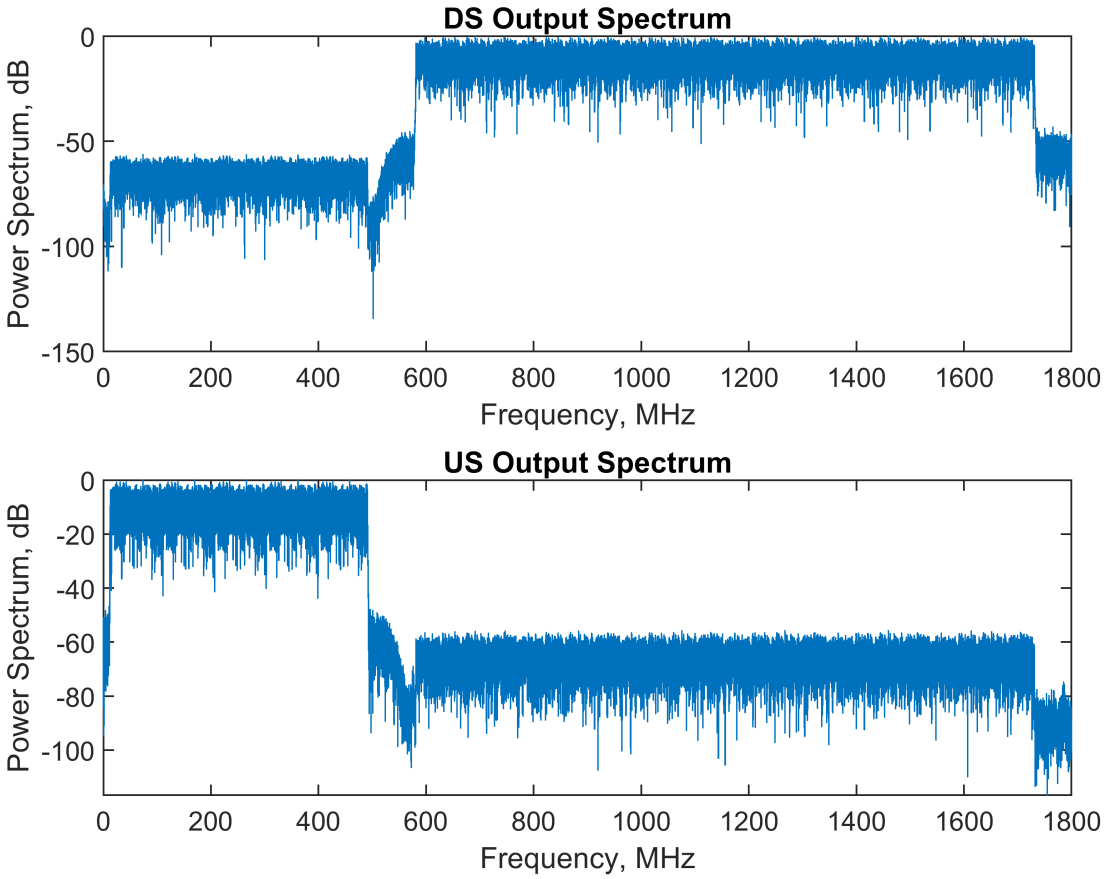


Figure 4.20: Test 3 - Upstream and downstream spectra at amplifier output (492/580 split)

Table 4.9: Test 3 - MER measurements for received channels

Upstream Channels		Downstream Channels	
Center Freq	MER	Center Freq	MER
60 MHz	42.05 dB	676 MHz	42.81 dB
156 MHz	42.05 dB	868 MHz	42.82 dB
252 MHz	42.06 dB	1060 MHz	42.80 dB
348 MHz	42.06 dB	1252 MHz	42.83 dB
444 MHz	42.03 dB	1444 MHz	42.82 dB
N/A	N/A	1636 MHz	42.81 dB

again demonstrated to achieve better than 41dB MER, which shows that the proposed system is able to successfully support 4096-QAM DOCSIS modulation. Since this is the most demanding modulation type required by DOCSIS, these results verify that the proposed amplifier is a promising alternative to conventional analog diplexer-based amplifiers for DOCSIS networks.

5. Conclusions

5.1 Summary

The demand for higher data rates from CATV service providers requires more bandwidth through HFC networks. By replacing the traditional diplex filters with digital diplex filters, the downstream and upstream signal bandwidths become programmable and adaptive, which means that more bandwidth can be obtained through digital diplex filters.

Implementing digital filters in a DOCSIS node involves significant challenges that must be addressed. A primary challenge includes finding the most appropriate sampling rates for the digital filters in the downstream and upstream paths. While higher sampling rate allows for larger bandwidth, digital filters require more multipliers and adders to adapt to the increasing sampling rate, which usually results in higher hardware cost. In this thesis, the sampling rates of the digital filters in the downstream and upstream paths were determined based on the bandwidth allocation defined by the DOCSIS standard. The selected sampling rates were used in Matlab and ModelSim simulations.

Another challenge involves selecting a particular filter implementation structure for the digital filters. A filter implementation structure shows how a signal propagates from the input port to the output port of a filter sample by sample. This thesis compared and contrasted various filter implementation structures in terms of their hardware costs. Further, suggestions were given to reduce the hardware costs of some proposed structures while maintaining a high transmission rate. Performance analysis of the selected structure was done using Matlab and ModelSim simulations.

5.2 Contributions

The primary contribution of this work is the development of the digital diplexer to be used for data transmission in a DOCSIS node. The digital diplexer is an extension of the traditional diplexer and it allows many designs that are impractical or impossible as traditional diplexer to work. The digital diplexer is composed of two digital filters, two directional couplers and two amplifiers. Each digital filter contains a set of coefficients which characterize the filter's response. We have shown how the filter's response changes as the filter coefficients change. The application of digital filters offers a powerful and unique benefit that allows the filter coefficients to be programmable during transmission.

Moreover, implementation guidelines for increasing data rates and reducing hardware costs were given. The trade-off between diplexer performance and hardware implementation complexity was evaluated through MATLAB and ModelSim simulations. Recommendations for the size and structure of the digital diplex filter were offered. Analysis revealed that these guidelines could help finding the balance between transmission rates and hardware costs of the digital diplexer.

5.3 Results and Conclusions

The main objective of this thesis is to design a hardware efficient digital diplexer for use in an HFC network. There are two transmission paths in a digital diplexer. One is called downstream path, while the other one is named upstream path. Each path contains a set of electronic devices. The sampling frequencies of the electronic devices in different paths are different, and can be optimized based on the DOCSIS 4.0 frequency division duplex spectrum options. The computation results showed that the downstream sampling frequency is 3588 MHz, and the upstream sampling frequency is 1616 MHz. Further, the frequency specifications of digital diplex filters were determined based on the frequency allocation defined by the DOCSIS standard. The starting point of a transition band could range from 85 MHz to 684 MHz, while the ending point is between 108 MHz and 808 MHz. Multiple filter implementation structures were compared and contrasted to find a structure that supports high sampling frequencies at the lowest hardware cost. After careful consideration, block-

based frequency domain filtering structure was selected and applied to the design.

Based on the filtering structure and parameters, a fixed point model of the digital diplexer was constructed in Verilog. A simulation was then conducted in ModelSim to verify the performance of the model in the FPGA development environment. Another fixed point model of the digital diplexer was built and tested in MATLAB. From the results of the simulations, we conclude that it is possible for a digital diplexer to work in the FPGA development environment. Moreover, based on the results of additional ModelSim and MATLAB simulations, we can conclude that the designed digital diplexer supports highest modulation orders allowed in DOCSIS (4096-QAM) and allows for dynamic switching of upstream/downstream transition point. We also investigated the possibility for digital diplex filters to achieve sharper transition bands by testing multiple filters with different sizes of transition band in MATLAB. The results showed that the size of a transition band can be decreased at the cost of higher hardware usage, which means that the 'wasted' bandwidth associated with higher split points can be reduced.

5.4 Future Work

The most relevant future work includes hardware testing with an actual FPGA in an actual lab containing DOCSIS equipment. The hardware performance of digital diplexers should be investigated to verify that the proposed scheme is able to support high speed transmission. Additional sources of noise in hardware such as phase noise will be taken into account in the experiments. Spectrum analyzers should be used to visualize the signals transmitted by the digital diplexers and cable modems can be used to demodulate the actual DOCSIS signals through the network.

Further research involves performance and resource usage. After testing with an actual digital diplexer, if the data rates are not high enough, a more powerful digital diplexer could be adapted to the DOCSIS network. If that turns out to be the case, increasing the size of the FFT processors in digital diplex filters is a possible solution worth considering, although it would cause an increase in hardware cost. Using a different filtering architecture such as FFA is also an alternative worth considering.

References

[1] California-Cable and Telecommunications-Association, "History of cable," 2015.

[2] E. S. Smith, "The emergence of CATV: A look at the evolution of a revolution," *Proceedings of the IEEE*, vol. 58, no. 7, pp. 967–982, 1970.

[3] J. S. Light, "Before the internet, there was cable," *IEEE Annals of the History of Computing*, vol. 25, no. 2, pp. 96–95, 2003.

[4] E. Smith, "Pilot two-way CATV systems," *IEEE Transactions on Communications*, vol. 23, no. 1, pp. 111–120, 1975.

[5] J. M. Cioffi, "Lighting up copper [history of communications]," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 30–43, 2011.

[6] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 1.0*, radio frequency interface specification ed., November 2001.

[7] P. Tungsakul, K. Songwatana, and P. Moungnuol, "A quality analysis of DOCSIS cable modem," in *2016 International Computer Science and Engineering Conference (ICSEC)*, pp. 1–6, 2016.

[8] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 1.1*, radio frequency interface specification ed., September 2005.

[9] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 2.0*, radio frequency interface specification ed., April 2009.

[10] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 3.0*, physical layer specification ed., December 2017.

[11] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 3.1*, physical layer specification ed., May 2018.

[12] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 4.0*, physical layer specification ed., August 2019.

[13] Cable Television Laboratories, *Data-Over-Cable Service Interface Specifications DOCSIS 3.1*, remote phy specification ed., May 2018.

[14] Electronics Notes, "Antenna RF diplexer," 2020.

[15] MACOM, *MAFL-011057 Datasheet*, CATV diplex filter: 5-85/102-1218 MHz ed.

[16] D. Large and J. Farmer, *Broadband Cable Access Networks: The HFC Plant*. Morgan Kaufmann, sixth ed.

[17] T. Cloonan, A. Al-Banna, F. O'Keeffe, J. Ulm, and R. Cloonan, "Capacity planning, traffic engineering, and HFC plant evolution for the next 25 years," in *2019 Fall Technical Forum for Society of Cable and Telecommunications Engineers-International Society of Broadband Experts, National Cable and Telecommunications Association and CableLabs*, 2019.

[18] J. T. Chapman and H. Jin, "FDX DOCSIS line extender: Deploying FDX DOCSIS beyond N+0," in *2018 Fall Technical Forum for SCTE-ISBE, NCTA and CableLabs*, 2018.

[19] W. Coomans and R. Coldren, "Full duplex DOCSIS over active (N+X) cable networks," in *2019 Fall Technical Forum for SCTE-ISBE, NCTA and CableLabs*, 2019.

[20] L. Litwin, "FIR and IIR digital filters," *IEEE Potentials*, vol. 19, no. 4, pp. 28–31, 2000.

[21] C. Erdoğan, I. Myderrizi, and S. Minaei, "FPGA implementation of BASK-BFSK-BPSK digital modulators [testing ourselves]," *IEEE Antennas and Propagation Magazine*, vol. 54, no. 2, pp. 262–269, 2012.

[22] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Pearson Higher Education Inc., third ed., 2010.

[23] H. H. Nguyen, "Introduction to OFDM," 2013.

[24] H. H. Nguyen and E. Shwedyk, *A First Course in Digital Communications*, pp. 302–342. Cambridge University Press, 2009.

[25] A. S. Kashi, J. C. Cartledge, A. Bakhshali, A. Rezania, A. I. A. El-Rahman, M. O'Sullivan, C. Laperle, A. Borowiec, and K. Roberts, "Information rates for the SP 128-QAM and DP 16-QAM modulation formats," in *2015 European Conference on Optical Communication (ECOC)*, pp. 1–3, 2015.

[26] J. O. Smith, *Introduction to Digital Filters with Audio Applications*, ch. Time Domain Digital Filter Representations. W3K Publishing, September 2007.

[27] M. Bellanger, *Digital Processing of Signals*. New York: John Wiley and Sons, 1984.

[28] T. Saramaki, T. Neuvo, and S. Mitra, "Design of computationally efficient interpolated FIR filters," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 1, pp. 70–88, 1988.

[29] J. McClellan and T. Parks, "A unified approach to the design of optimum FIR linear-phase digital filters," *IEEE Transactions on Circuit Theory*, vol. 20, no. 6, pp. 697–701, 1973.

[30] D. Parker and K. Parhi, "Area-efficient parallel FIR digital filter implementations," in *Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP '96*, pp. 93–111, 1996.

[31] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, ch. Algorithmic Strength Reduction in Filters and Transforms, pp. 255–296. Wiley-Interscience, January 1999.

[32] M. Narasimha, "Modified overlap-add and overlap-save convolution algorithms for real signals," *IEEE Signal Processing Letters*, vol. 13, no. 11, pp. 669–671, 2006.

[33] L. Yu, "Matlab programming environment based on web," in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 509–512, 2018.

[34] B. Gestner and D. V. Anderson, "Automatic generation of modelsim-matlab interface for RTL debugging and verification," in *2007 50th Midwest Symposium on Circuits and Systems*, pp. 1497–1500, 2007.

[35] Intel, *FFT IP Core User Guide*, updated for intel quartus prime design suite: 17.1 ed., November 2017.

[36] G. White, K. Sundaresan, and B. Briscoe, "Low latency DOCSIS overview and performance characteristics," in *2019 Fall Technical Forum for SCTE-ISBE, NCTA and CableLabs*, 2019.

A. Defining FFT IP Core Signals

FFT IP core interfaces define a standard, pliable, and modular protocol for data transmissions from a source interface to a sink interface [35]. The following tables show the direction of every FFT IP core signal and briefly explain the purpose of each signal.

Table A.1: FFT IP core signals - Part 1

Signal Name	Direction	Description
clk	Input	Clock signal that clocks all internal FFT engine components.
reset_n	Input	Active-low asynchronous reset signal.
sink_eop	Input	Indicates the end of the incoming FFT frame.
sink_error	Input	Indicates an error has occurred in an upstream module due to an illegal usage of the Avalon-ST protocol.
sink_imag	Input	Imaginary input data, which represents a signed number of data precision bits.
sink_ready	Output	Indicates the FFT engine can accept data.
sink_real	Input	Real input data, which represents a signed number of data precision bits.
sink_sop	Input	Indicates the start of the incoming FFT frame.
sink_valid	Input	Asserted when data on the data bus is valid.

Table A.2: FFT IP core signals - Part 2

Signal Name	Direction	Description
source_eop	Output	Marks the end of the outgoing FFT frame.
source_error	Output	Indicates an error has occurred either in an upstream module or within the FFT module.
source_exp	Output	This exponent accounts for scaling of internal signal values during FFT computation.
source_imag	Output	Imaginary output data.
source_ready	Input	Indicates the downstream module can accept data.
source_real	Output	Real output data.
source_sop	Output	Marks the start of the outgoing FFT frame.
source_valid	Output	Asserted by the FFT when there is valid data to output.
inverse	Input	Asserted when inverse FFT should be computed.

B. Complete Amplifier Test Simulation Results

The following figures show the frequency spectra of the upstream and downstream signals that were generated in MATLAB and passed into the digital diplexer for 300 MHz, 492 MHz and 684 MHz upstream cases. The constellation diagrams of the received signals in each test are also available in this appendix.

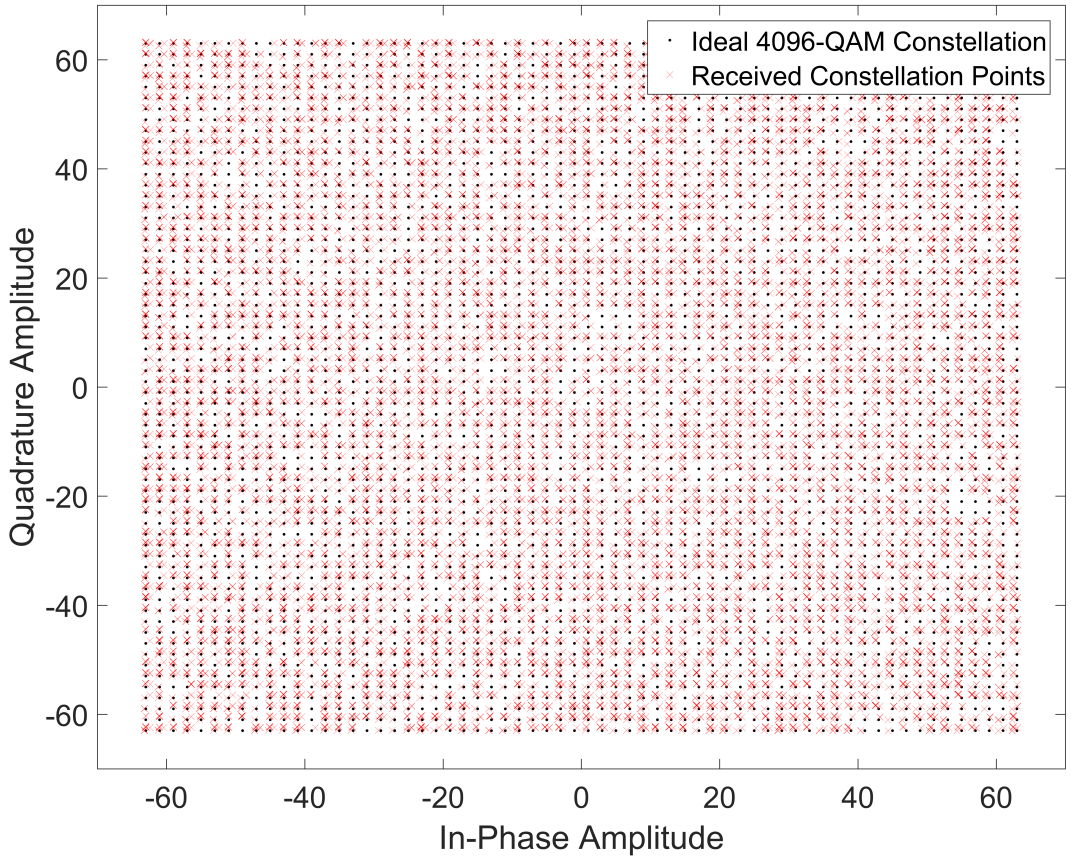


Figure B.1: Test 1 - received signal constellation for downstream channels (300/354 split)

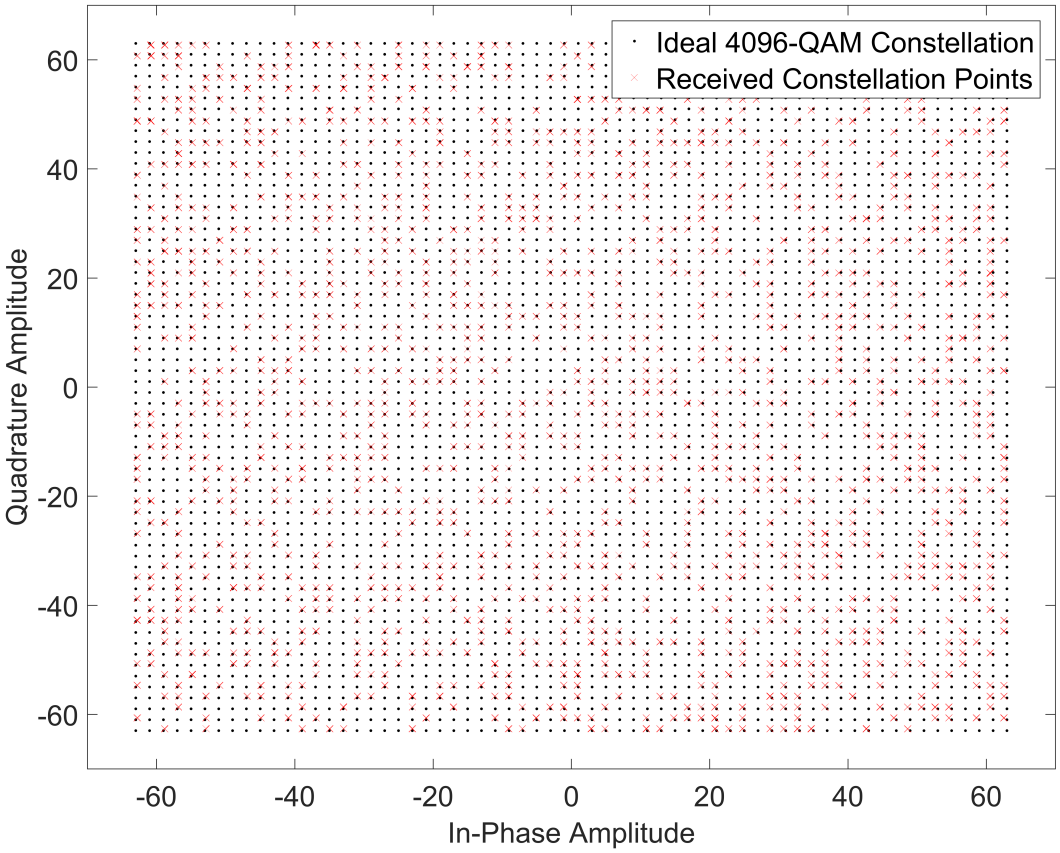


Figure B.2: Test 1 - received signal constellation for upstream channels (300/354 split)

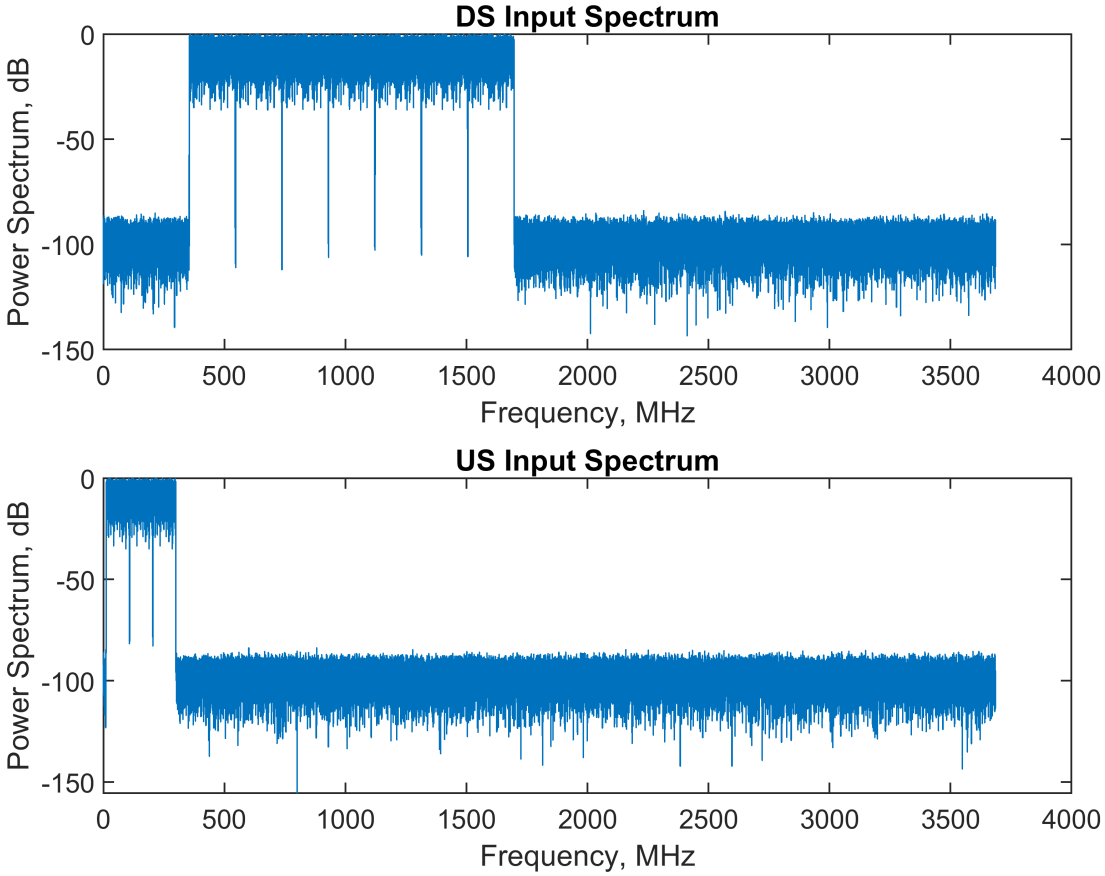


Figure B.3: Test 1 - Upstream and downstream spectra at amplifier input (300/354 split)

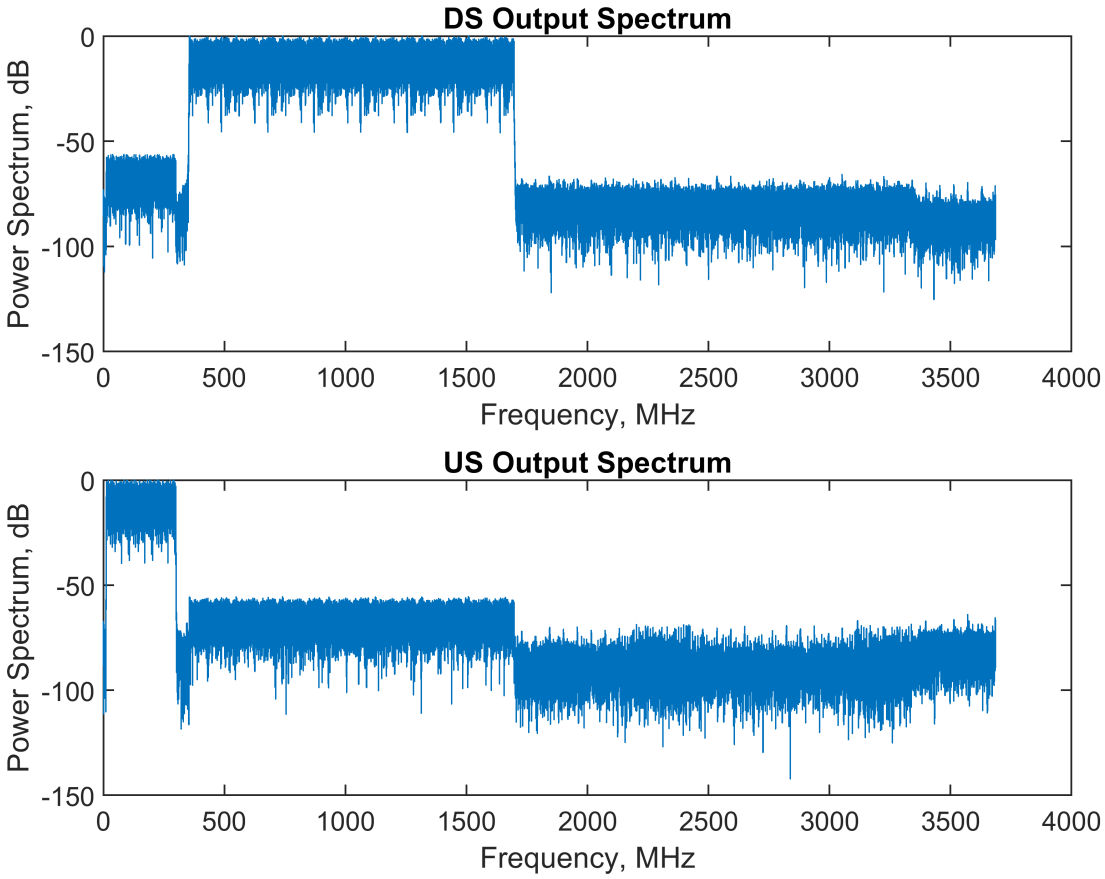


Figure B.4: Test 1 - Upstream and downstream spectra at amplifier output (300/354 split)

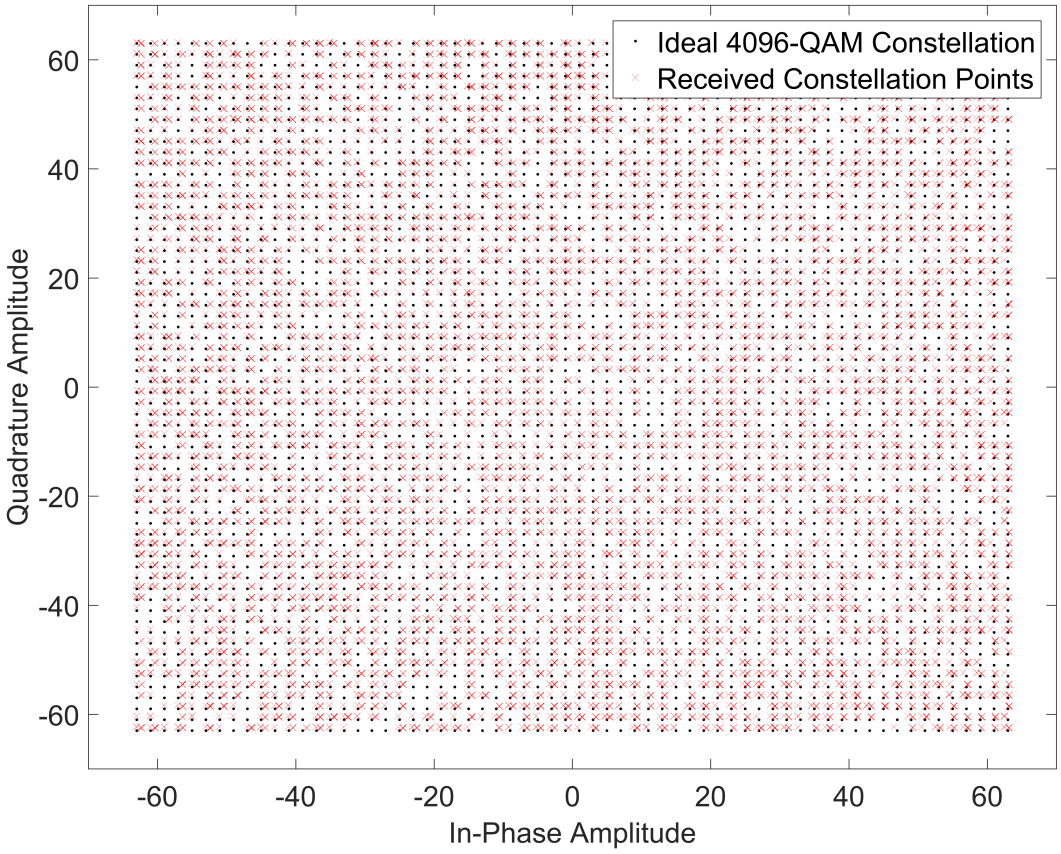


Figure B.5: Test 1 - received signal constellation for downstream channels (492/580 split)

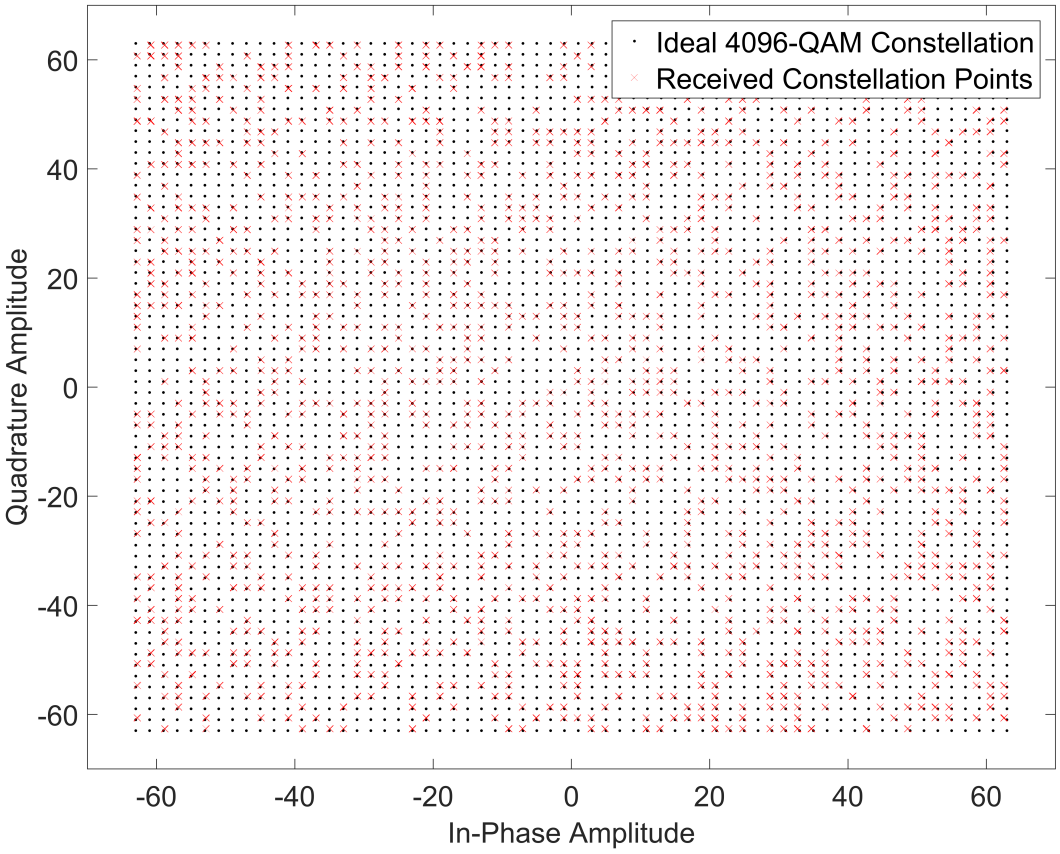


Figure B.6: Test 1 - received signal constellation for upstream channels (492/580 split)

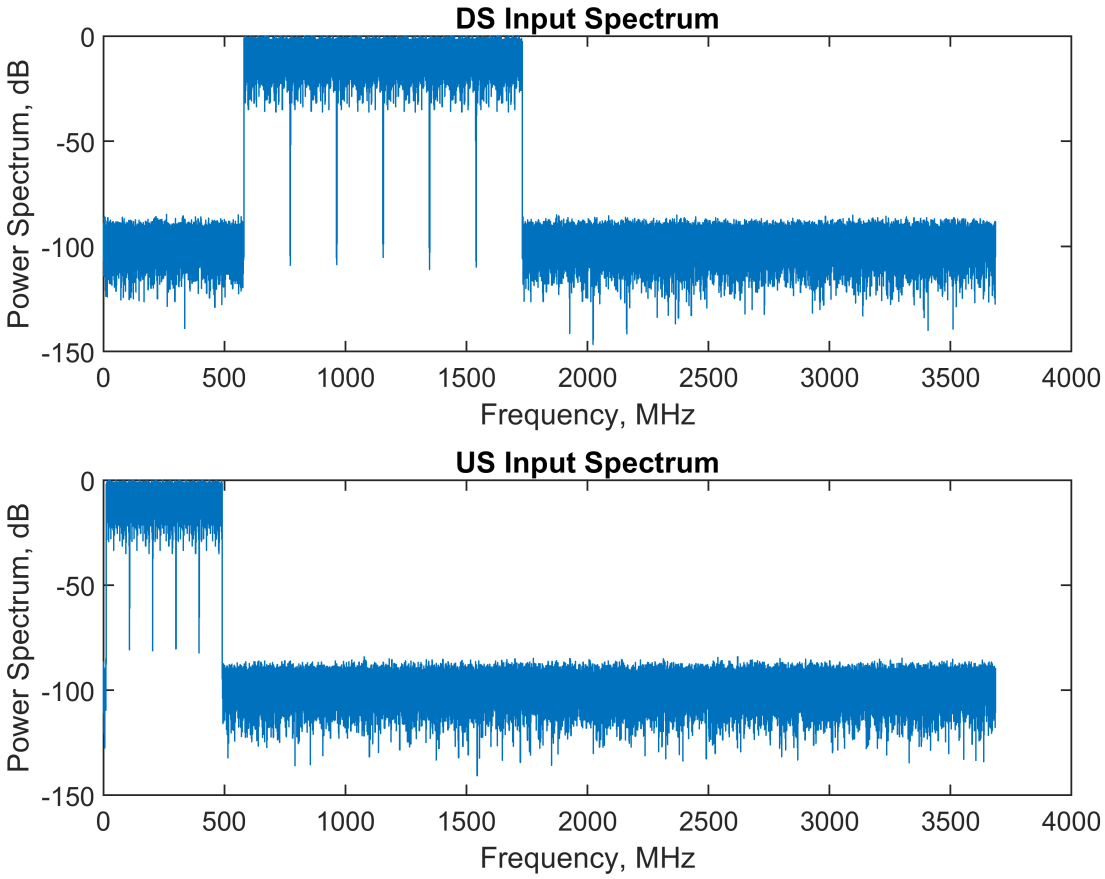


Figure B.7: Test 1 - Upstream and downstream spectra at amplifier input (492/580 split)

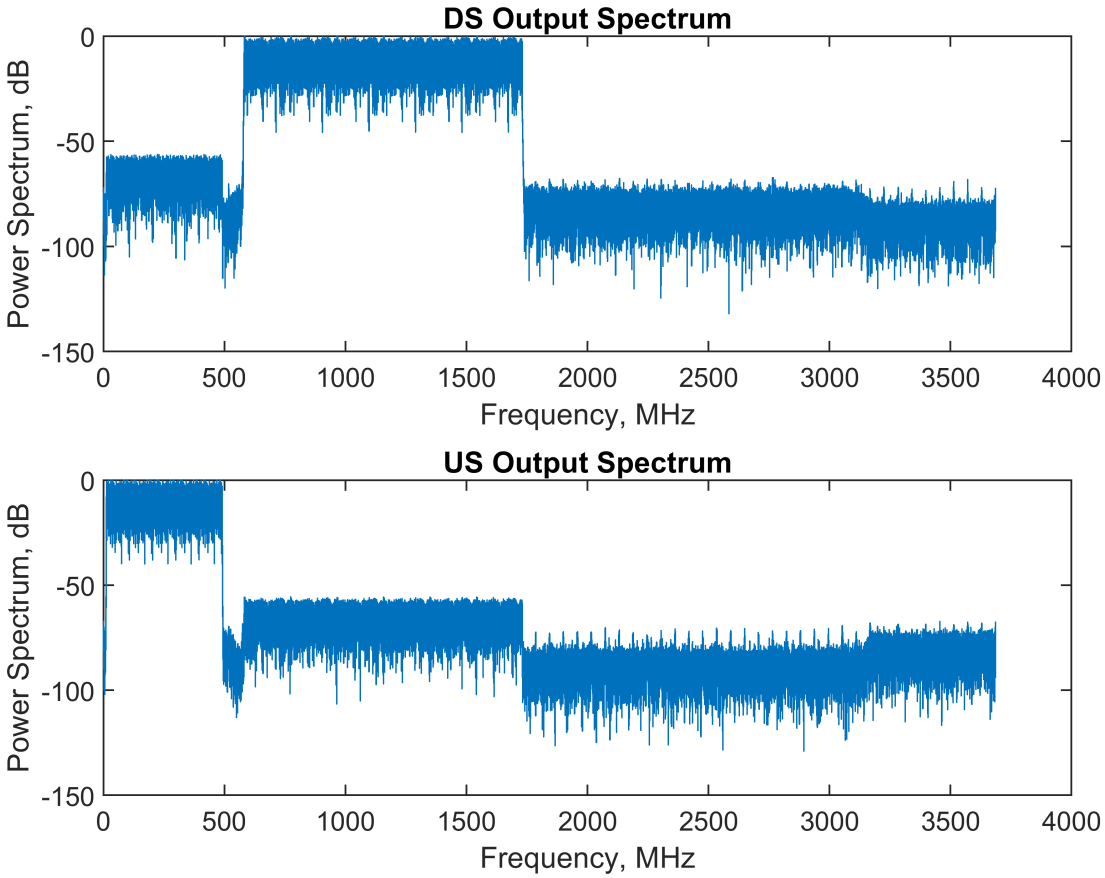


Figure B.8: Test 1 - Upstream and downstream spectra at amplifier output (492/580 split)

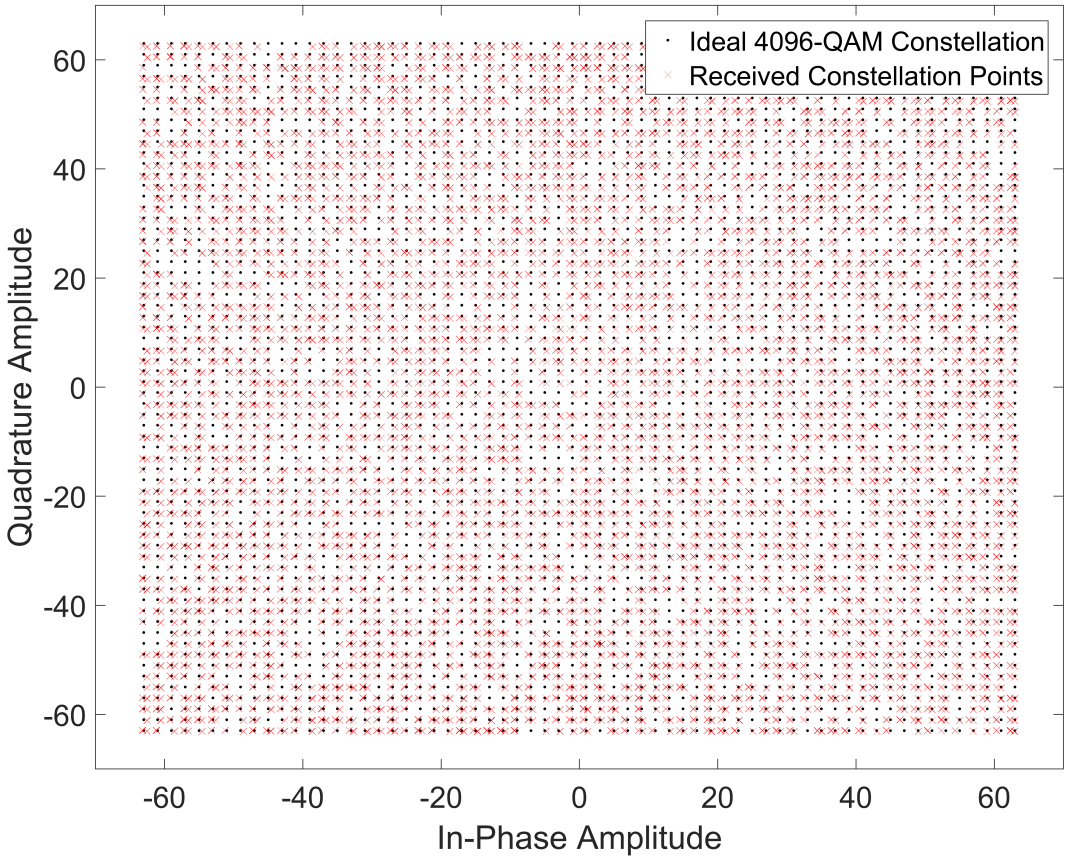


Figure B.9: Test 1 - received signal constellation for downstream channels (684/808 split)

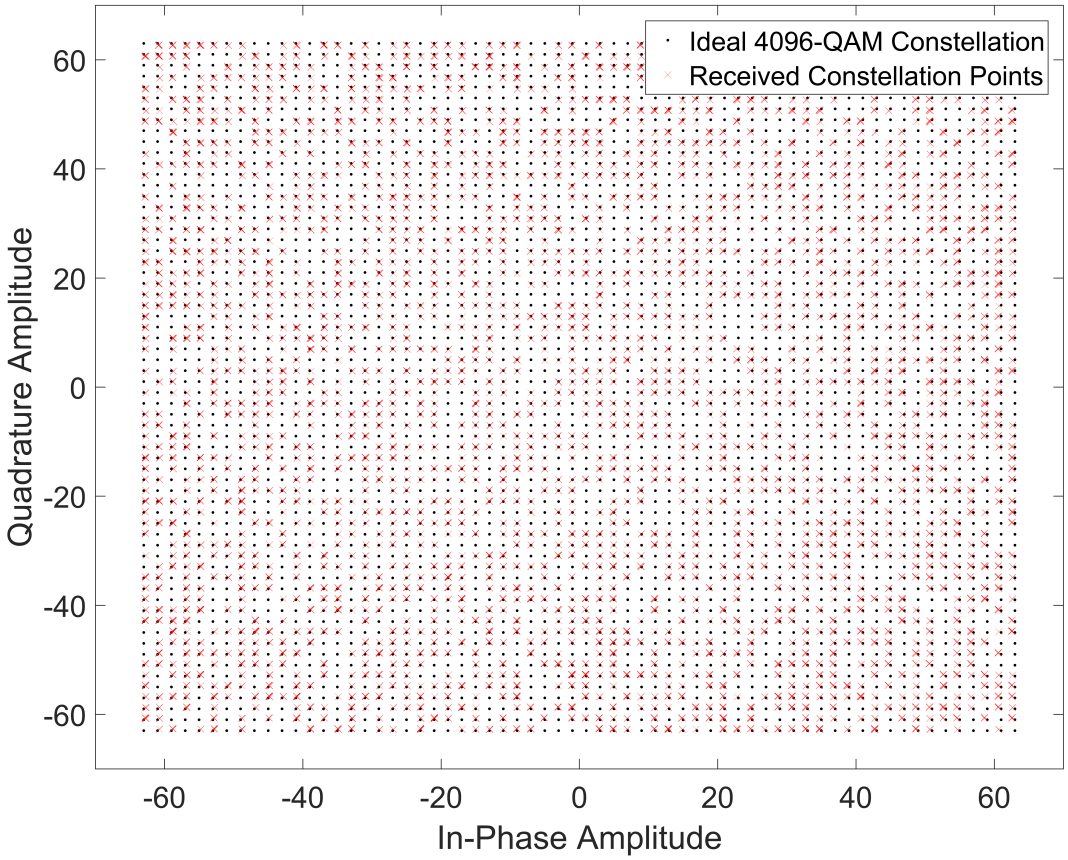


Figure B.10: Test 1 - received signal constellation for upstream channels (684/808 split)

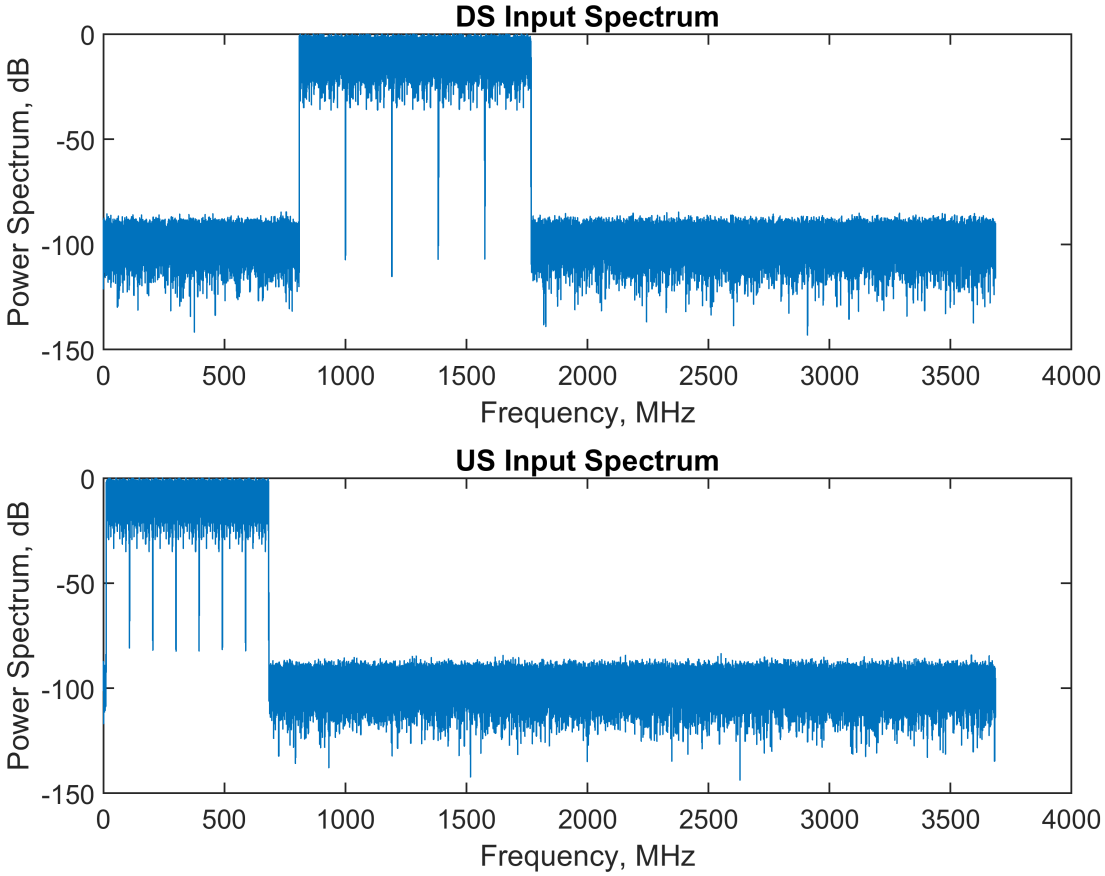


Figure B.11: Test 1 - Upstream and downstream spectra at amplifier input (684/808 split)

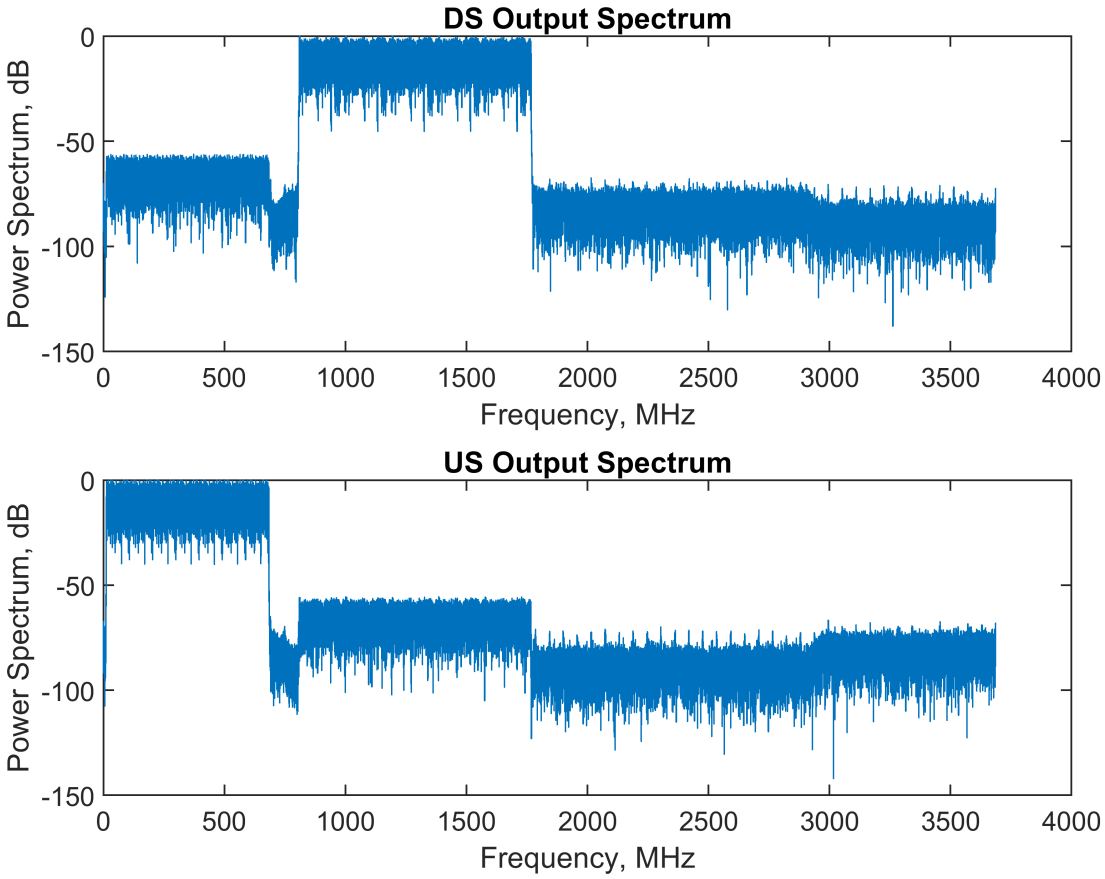


Figure B.12: Test 1 - Upstream and downstream spectra at amplifier output (684/808 split)

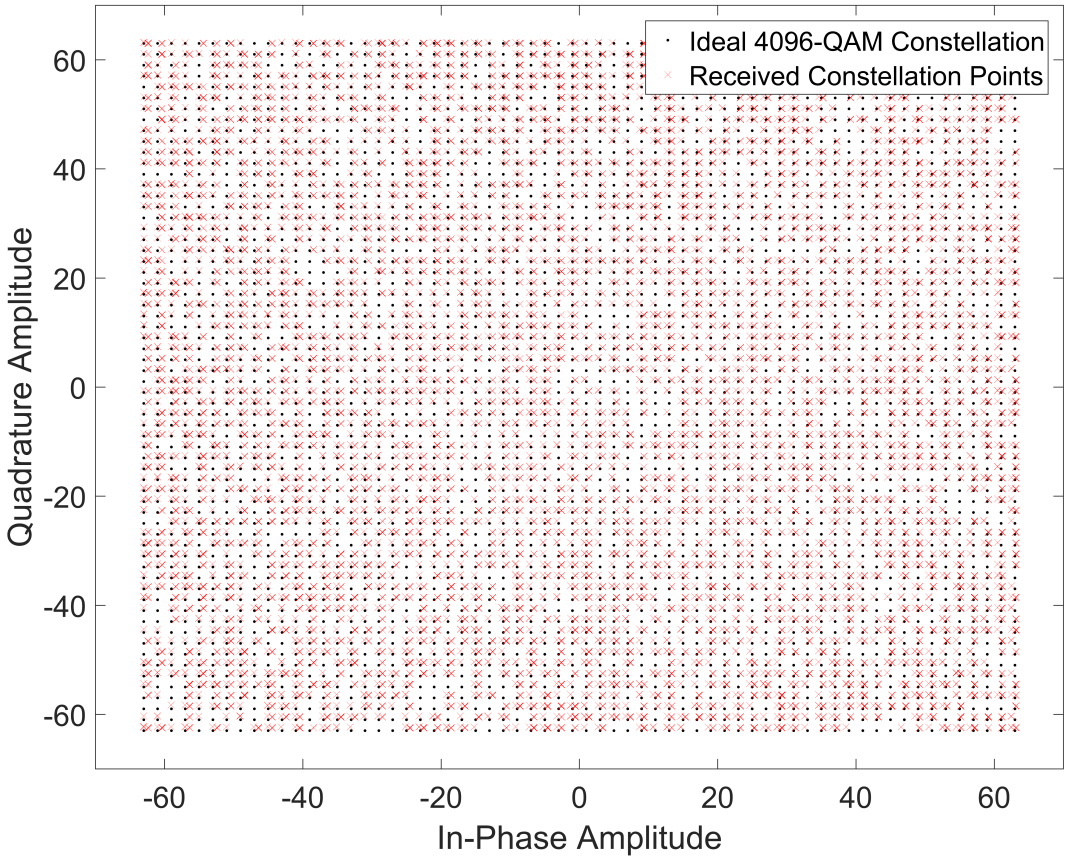


Figure B.13: Test 2 - received signal constellation for downstream channels

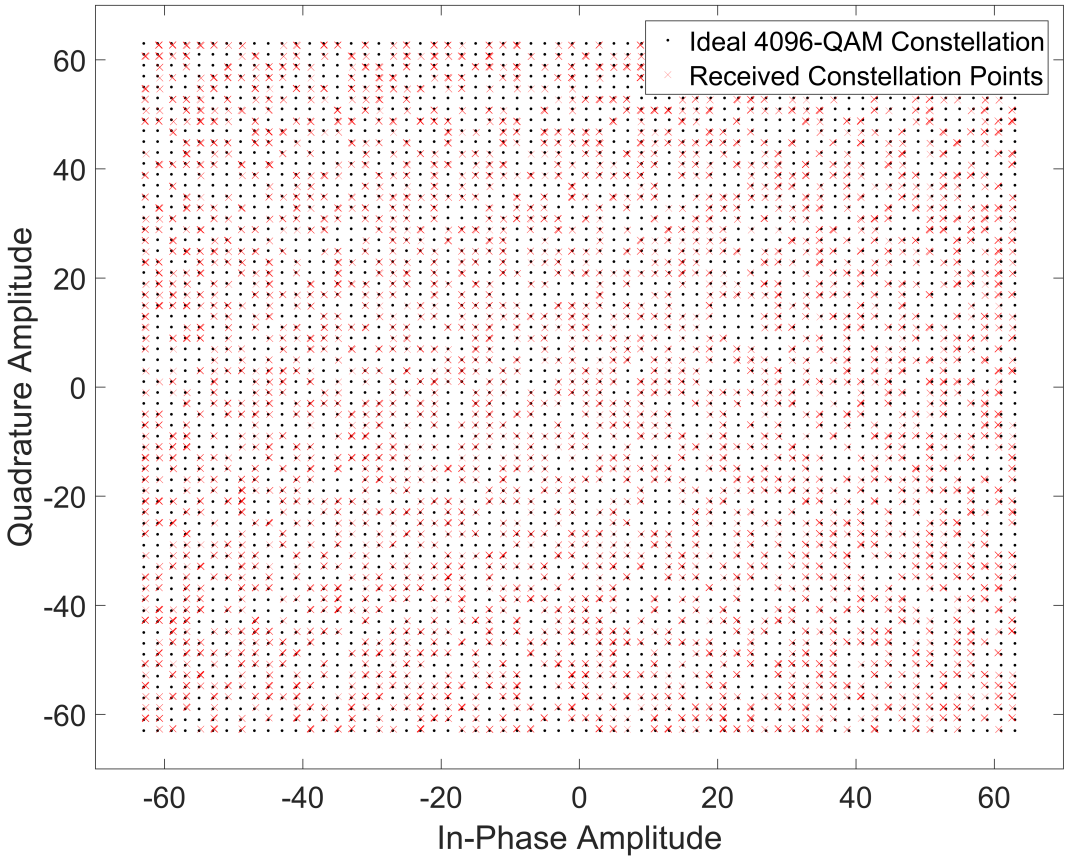


Figure B.14: Test 2 - received signal constellation for upstream channels

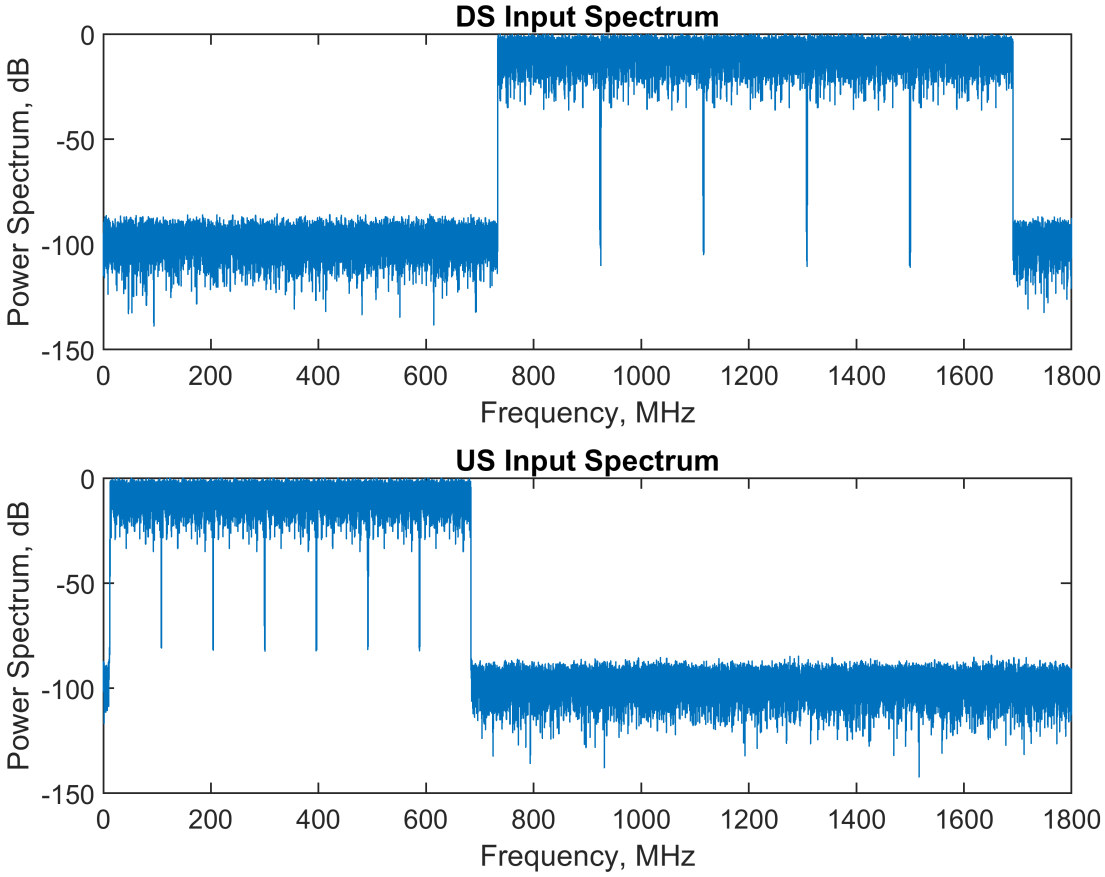


Figure B.15: Test 2 - Upstream and downstream spectra at amplifier input

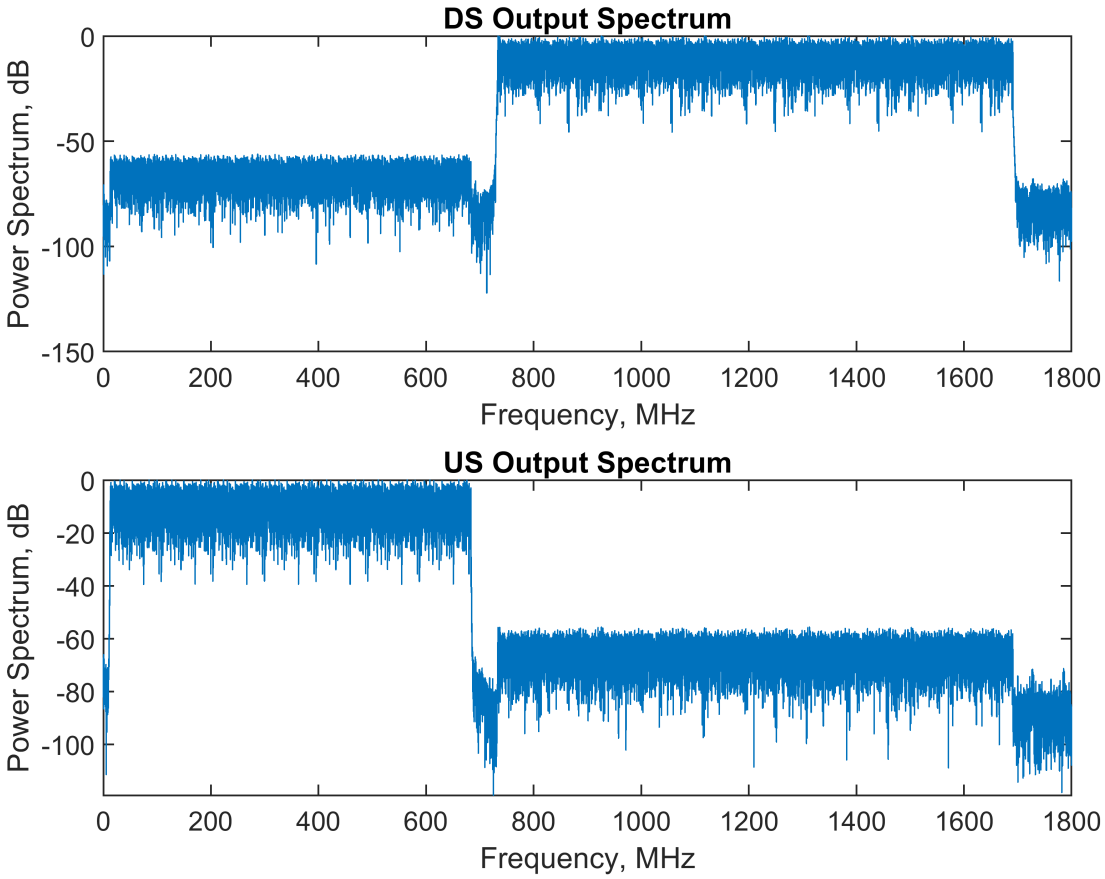


Figure B.16: Test 2 - Upstream and downstream spectra at amplifier output

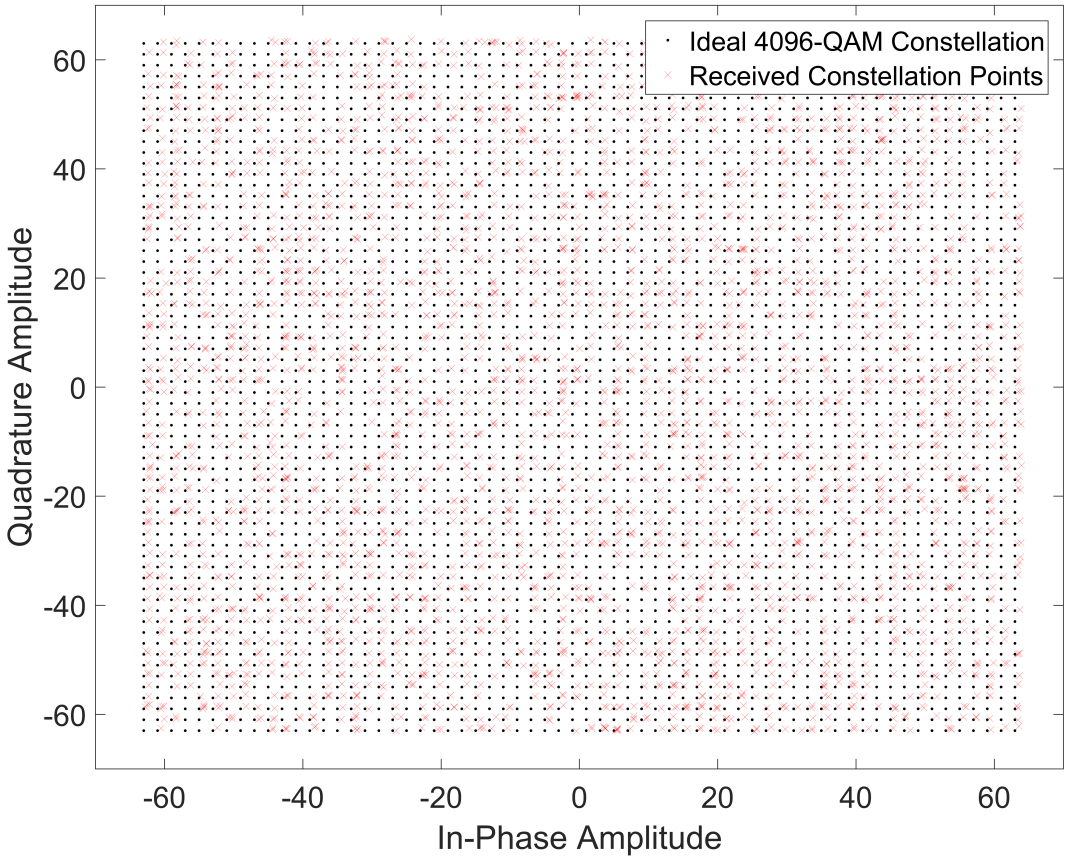


Figure B.17: Test 3 - received signal constellation for downstream channels

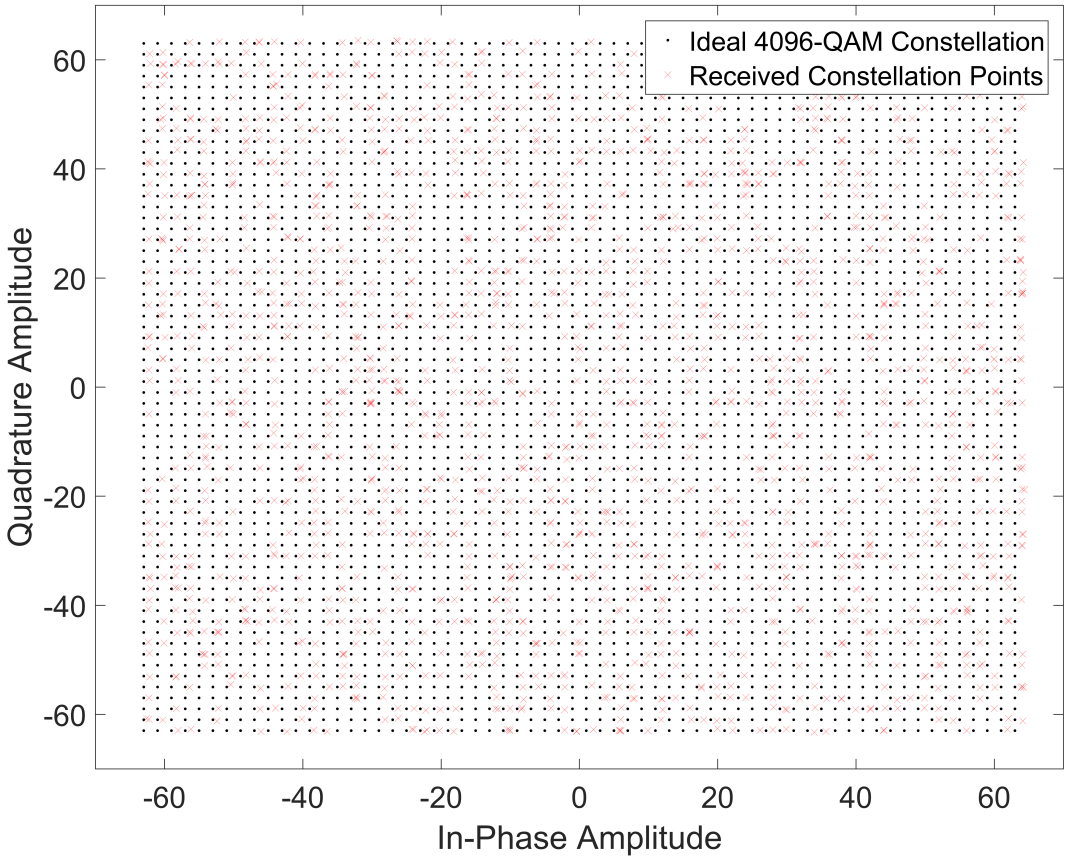


Figure B.18: Test 3 - received signal constellation for upstream channels

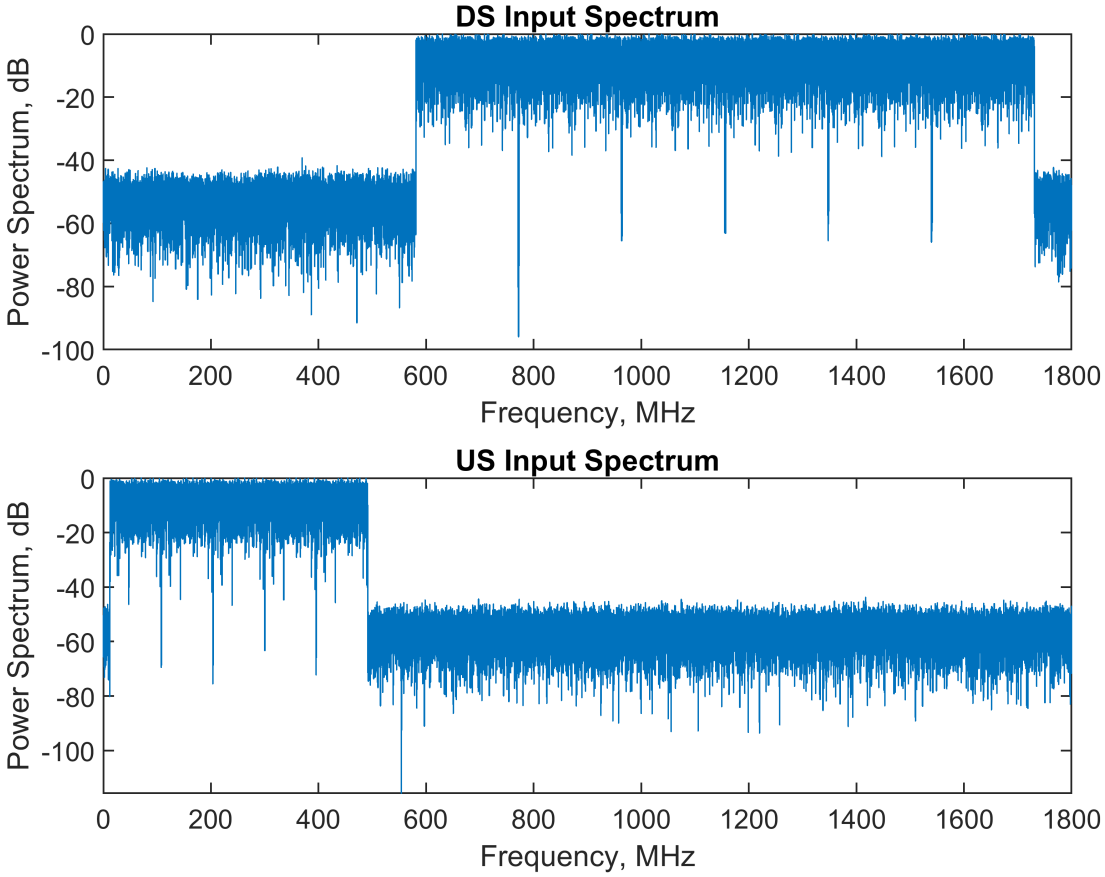


Figure B.19: Test 3 - Upstream and downstream spectra at amplifier input

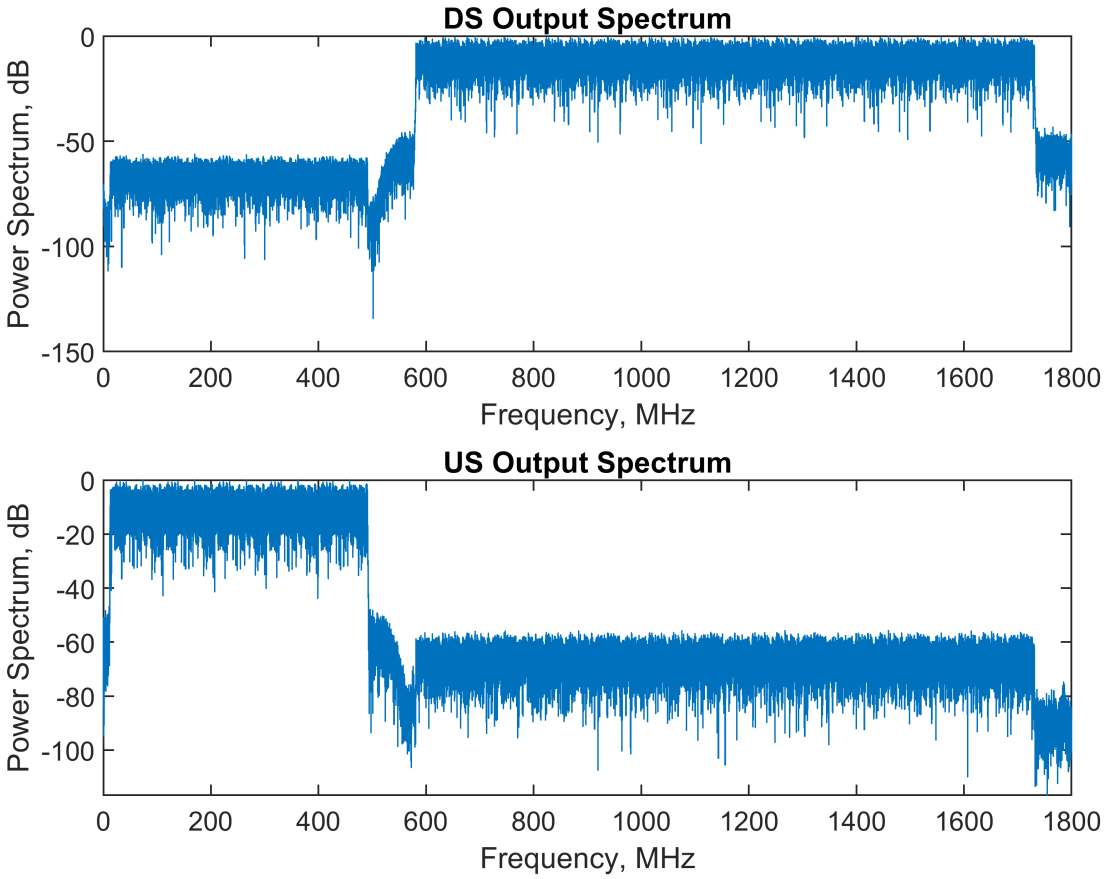


Figure B.20: Test 3 - Upstream and downstream spectra at amplifier output