DIGITAL IMPLEMENTATION OF AN UPSTREAM DOCSIS QAM MODULATOR AND CHANNEL EMULATOR

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

by

Andy D. Fontaine

Saskatoon, Saskatchewan, Canada
September 2014

Permission to Use

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

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

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

Acknowledgments

I would like to thank my supervisors, Dr. Eric Salt and Dr. Ha Nguyen for their patience and encouragement throughout the duration of the program. Their courses were well structured and provided me with practical tools which helped to cement my knowledge during the project development phase. Further, Dr. Salt's guidance during the writing process was very helpful and something for which I am enormously grateful.

A very special thanks goes to David Dodds and Brian Daku for steering me towards the Industry Oriented Master's (IOM) when I first inquired about the graduate programs being offered by the department. The financial assistance which Mr. Dodds was able to provide was of tremendous benefit, and I am extremely fortunate and appreciative.

I would also like to thank my fellow IOM cohorts Mushtafizur Choudhury and Rory Gowen. Their ideas and suggestions have proved very useful and helped to guide the development of this project. In addition, Mr. Gowen created some valuable software that allowed me to convert RAM output data for circuit performance analysis in MATLAB. His contribution was instrumental to the performance evaluation of the channel emulator.

Finally, I must thank my parents, Dennis and Bette, whose enduring love, patience, and support have made this journey possible. Their faith in my abilities kept me on track whenever I was doubting myself. Thank you so much — I could not have done this without you.

Abstract

The concept of cable television, originally called community antenna television (CATV), began in the 1940's. The information and services provided by cable operators have changed drastically since the early days. Cable service providers are no longer simply providing their customers with broadcast television but are providing a multi-purpose, two-way link to the digital world. Custom programming, telephone service, radio, and high-speed internet access are just a few of the services offered by cable service providers in the 21st century.

At the dawn of the internet the dominant mode of access was through telephone lines. Despite advances in dial-up modem technology, the telephone system was unable to keep pace with the demand for data throughput. In the late 1990's an industry consortium known as Cable Television Laboratories, Inc. developed a standard protocol for providing high-speed internet access through the existing CATV infrastructure. This protocol is known as Data Over Cable Service Interface Specification (DOCSIS) and it helped to usher in the era of the information superhighway.

CATV systems use different parts of the radio frequency (RF) spectrum for communication to and from the user. The downstream portion (data destined for the user) consumes the bulk of the spectrum and is located at relatively high frequencies. The upstream portion (data destined to the network from the user) of the spectrum is smaller and located at the low end of the spectrum. This lower frequency region of the RF spectrum is particularly prone to impairments such as micro-reflections, which can be viewed as a type of multipath interference. Upstream data transfer in the presence of these impairments is therefore problematic and requires complex signal correction algorithms to be employed in the receiver.

The quality of a receiver is largely determined by how well it mitigates the signal impairments introduced by the channel. For this reason, engineers developing a receiver require a piece of equipment that can emulate the channel impairments in any permutation in order to test their receiver. The conventional test methodology uses a hardware RF channel emulator connected between the transmitter and the receiver under test. This method not only requires an expensive RF channel emulator, but a functioning analog front-end as well. Of

these two problems, the expense of the hardware emulator is likely less important than the delay in development caused by waiting for a functional analog front-end. Receiver design is an iterative, time consuming process that requires the receiver's digital signal processing (DSP) algorithms be tested as early as possible to reduce the time-to-market.

This thesis presents a digital implementation of a DOCSIS-compliant channel emulator whereby cable micro-reflections and thermal noise at the analog front-end of the receiver are modelled digitally at baseband. The channel emulator and the modulator are integrated into a single hardware structure to produce a compact circuit that, during receiver testing, resides inside the same field programmable gate array (FPGA) as the receiver. This approach removes the dependence on the analog front-end allowing it to be developed concurrently with the receiver's DSP circuits, thus reducing the time-to-market.

The approach taken in this thesis produces a fully programmable channel emulator that can be loaded onto FPGAs as needed by engineers working independently on different receiver designs. The channel emulator uses 3 independent data streams to produce a 3-channel signal, whereby a main channel with micro-reflections is flanked on either side by adjacent channels. Thermal noise normally generated by the receiver's analog front-end is emulated and injected into the signal. The resulting structure utilizes 43 dedicated multipliers and 401.125 KB of RAM, and achieves a modulation error ratio (MER) of 55.29 dB.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACI Adjacent Channel Interference

APF All Pass Filter

AWGN Additive White Gaussian Noise

BASK Binary Amplitude Shift Keying

BPSK Binary Phase Shift Keying

CATV Community Antenna TeleVision

CM Cable Modem

CMTS Cable Modem Termination System

COI Channel of Interest (also referred to as the main channel)

CORDIC COordinate Rotation DIgital Computer

DAC Digital to Analog Converter

dB DeciBels

dBc DeciBels with respect to Carrier

DDS Direct Digital Synthesis

DEMUX Demultiplexer

DFT Discrete Fourier Transform

DOCSIS Data Over Cable Service Interface Specification

FDMA Frequency Division Multiple Access

FIFO First-In First-Out

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

I In-phase (Component of Quadrature Modulated Signal)

IIR Infinite Impulse Response

IP Intellectual Property

ISI InterSymbol Interference

JTAG Joint Test Action Group

LED Light Emitting Diode

LFSR Linear Feedback Shift Register

LPF Low-Pass Filter

LSB Least Significant Bit

LUT Look Up Table

MAC Media Access Control *or* Multiply and ACcumulate - obvious from context

MATLAB MATrix LABoratory

MER Modulation Error Ratio

MIF Memory Initialization File

MSB Most Significant Bit

MSE Mean Squared Error

MUX Multiplexer

NCO Numerically Controlled Oscillator

Q Quadrature (Component of Quadrature Modulated Signal)

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RAM Random Access Memory

ROM Read Only Memory

RF Radio Frequency

S-CDMA Synchronous Code Division Multiple Access

SNR Signal to Noise Ratio

SRRC Square Root Raised Cosine

TDMA Time Division Multiple Access

VSA Vector Signal Analyzer

XOR eXclusive OR

1. Introduction

Today's cable television industry is an outgrowth of the work of a few enterprising inventors in the late 1940's. Although there is some disagreement as to who built the very first cable television system, L. E. Parsons of Astoria, Oregon and John Walson, Sr. of Mahanoy City, Pennsylvania are credited with independently constructing the first cable television networks [1], [2]. Located on opposite ends of the United States, these men pioneered a new way of delivering broadcast television to a much wider audience. Areas that are remote or located in mountainous terrain do not have a line of sight to the transmitter and signal reception is severely degraded. By building a large antenna in a location with a line of sight to the transmitter, the received radio frequency (RF) signal could be amplified and passed through coaxial cables into the televisions of those in the community. Originally known as community antenna television (CATV), these networks have evolved substantially over the course of the last few decades.

Cable television was initially developed to produce TV reception for viewers in areas with poor reception, as depicted in Figure 1.1. These viewers were required to pay a connection fee to be granted access to the cable. As its popularity grew, entrepreneurs saw opportunities to invest in a burgeoning industry. It was found that viewers were willing to pay a subscription fee for the reception of television broadcasts, and the notion of pay TV was born. Several companies such as Zenith Radio Corporation, the International Telemeter Corporation, and Skiatron were active throughout the 1950's and 60's developing strategies to provide metered 'pay-as-you-see' content to cable TV subscribers [3]. Although the services developed by these companies were experimental and did not see widespread use, they helped to identify not only the types of services customers expected, but how these services could be effectively

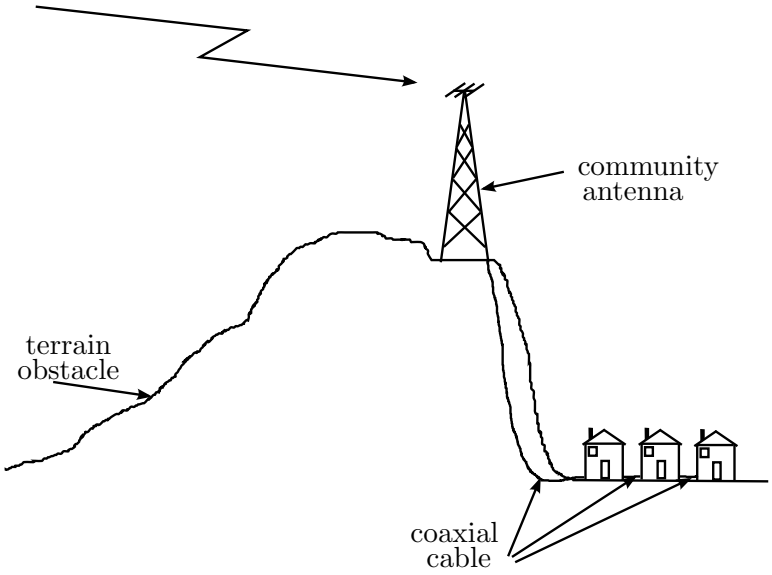


Figure 1.1: Example of Community Antenna Television

provided.

With the advent of satellite technology, the cable television industry underwent another drastic change in the 1970's. Satellites could deliver programming to cable systems, allowing more channels than was possible from transmitters on the ground [4]. With its high channel content, this type of cable television allowed the notion of pay TV to gain traction with an ever larger market. Cable television was no longer simply a way for remote viewers to tune into regular broadcast content. It had become a means to deliver a premium television service to viewers anywhere. For viewers who were already able to receive over-the-air broadcast signals, cable TV had evolved into a high quality service worth paying for.

Aside from satellite advances, companies such as Vicom and CAS Manufacturing were developing methods to provide a true two-way cable service [5]. By the 1980's, the original one-way service provided by cable companies was being replaced by a two-way system that allowed cable service providers to provide pay-per-view content to their customers. Orders could be placed through a set-top box in the subscriber's home, sending commands through the coaxial cable network to the head-end. Such systems paved the way for the modern two-way digital cable communications networks currently in use. At the time, however, the internet was still in its infancy, and dial-up modems were the dominant data transfer

technology. The birth of the worldwide web at the turn of the 1990's once again changed the game [6], as it became another profitable telecommunication service with a high demand. The complexity of web pages and the amount of data they contained quickly grew, and despite advances in dial-up modem technology, telephone-based internet became inadequate for many subscribers.

Fortunately, a solution to the rapidly expanding demands for bandwidth was already in place. By the middle of the 1990's, cable networks were well established in many areas [7], [8], and technological advances in signal processing allowed the same two-way functionality enjoyed by telephone networks [9]. These cable networks were able to provide a wide range of services through a single coaxial connection. Further, the higher bandwidths available on CATV networks allow data transmission at much higher rates than was previously possible with dial-up modems, making them the preferred technology for the emerging information age.

Although cable networks were in widespread use and were a capable medium for data transfer, there was no standardized method of transferring data over the CATV infrastructure. Cable service providers were generally forced to purchase entire networks or systems from a single manufacturer. Among other standards developed in the late 1990's [10], an open-source specification was developed by an industry consortium known as Cable Television Laboratories, Inc. (CableLabs), with the intent of standardizing high-speed data over cable service. This standard, the Data Over Cable Service Interface Specification (DOCSIS), provides a complete framework for the types of systems to be used from head-end to customer as well as specific tolerances for conditions on the network. It allows interoperability between components manufactured by different companies while maintaining a high quality of service under ever-changing conditions.

Current CATV networks use duplexers and bidirectional amplifiers to allow two-way communication on the cable. The RF spectrum is partitioned into two bands: downstream (from head-end to user) and upstream (from user to head-end). The wide array of services consumes the available downstream bandwidth, so in order to maximize the data throughput to the customer, the spectrum is divided according to demand. A relatively small portion

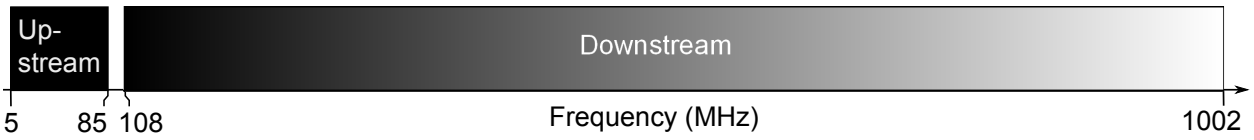


Figure 1.2: DOCSIS 3.0 Extended RF Spectrum

of bandwidth at the low end of the spectrum is allotted to the upstream direction, and the remainder is allotted to the downstream direction. In the DOCSIS 3.0 physical layer specification the upstream spectrum is either 5–30 MHz, 5–42 MHz, or the 5–85 MHz extended spectrum. The lower edge of the downstream spectrum is either 54 or 108 MHz. The upper edge is implementation dependent and can be anywhere from 300–1002 MHz [11]. This is shown in Figure 1.2 for the extended upstream spectrum.

It is clear that downstream data transfer is much easier than in the upstream direction. The first reason is obvious — there is more bandwidth available for simultaneous transmission of many more channels. The second reason is that the lower end of the RF spectrum is an especially noisy region [12]. Historically, this was not a major issue since upstream data traffic for pay-per-view used robust, low-rate, low-order modulation schemes [13]. However, the upstream data traffic in two-way internet communications require high modulation orders to achieve a high data throughput, causing difficulties with efficient data transfer.

Although CATV networks use shielded coaxial cables and are not as prone to interference as wireless networks, they are still subject to imperfections that can pose significant receiver design challenges. For example, spurious emissions from adjacent channels can spill over into the channel of interest (COI) and negatively impact signal quality. The signal located in the COI can also be subjected to echoes which cause the signal to interfere with itself. This phenomenon, known as multipath, is a formidable opponent for a receiver. A discussion of the causes and effects of multipath will be deferred until later.

Thermal noise generated at the analog front-end of a receiver is an impairment that cannot be avoided and care must be taken to ensure signals are transmitted with sufficient power. However, boosting the signal power with impunity is costly both in terms of hardware and power consumption. A balance between signal power and noise must be achieved, and

the aim is to set the transmit power just high enough to dominate over the noise.

Ingress is also a factor in CATV networks, particularly in the upstream direction. Ingress refers to signals emitted from transmitters that are not a part of the CATV network. The upstream path of a CATV network consists of a large sum of modem signals combined into a common cable trunk. Any shielding flaw in the cable connecting each modem to the trunk can cause the cable to act as an antenna and capture a portion of the ingress signal. With potentially thousands of modems interconnected over a wide geographical area, significant ingress regions can form in the upstream spectrum. Recurring narrowband ingress can be easily avoided, but sporadic ingress is much more difficult to avoid.

Before discussing the performance measures used for CATV channels, the meaning of data symbols and average symbol energy must be clarified. Consider first a serial bit sequence. Every bit in the sequence is in one of two possible states: 1 or 0. The bits in this binary sequence can be encoded in the amplitude of a real signal. For example, the values $+1$ and -1 could be used for 1 and 0 respectively. This type of modulation is commonly referred to as binary phase shift keying (BPSK). The binary data has been encoded into symbols — unique signal locations allocated to each combination of bits in the data. Multiple data bits can be encoded by creating more discrete amplitudes for mapping. For example, 2-bit symbols could be mapped to the amplitudes $-3/4$, $-1/4$, $+1/4$, and $+3/4$ — a type of modulation known as binary amplitude shift keying (BASK).

Complex values are frequently used to map symbols in a plane rather than on a line, which is known as quadrature amplitude modulation (QAM). The real component is referred to as the in-phase (I) component and the imaginary component is referred to as the quadrature (Q) component. The set of possible mapping locations is often referred to as a constellation. The average symbol energy, E_{av}, is given by

$$E_{av} = \frac{\sum_{k=1}^{K} I_k^2 + Q_k^2}{K}, \tag{1.1}$$

where K is the number of symbol locations in the constellation, and I_k and Q_k are the in-phase and quadrature components of the k-th symbol in the constellation. For example,

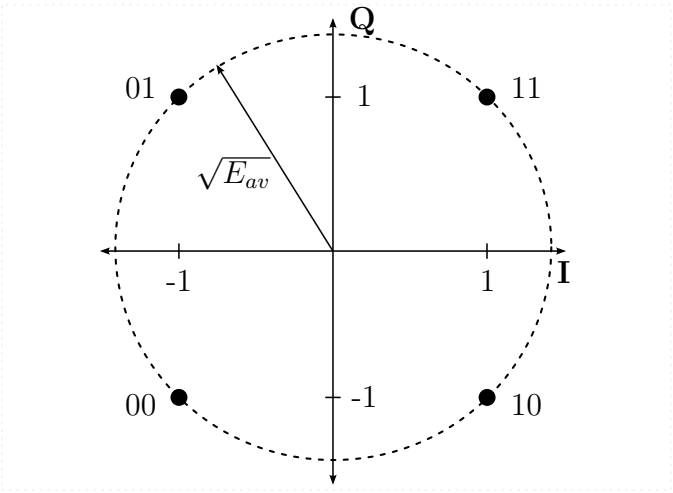


Figure 1.3: Example of QPSK Symbol Constellation

a 4-QAM constellation (also known as quadrature phase shift keying (QPSK)) has 4 unique symbol locations as depicted in Figure 1.3. If these are located at $s_1 = (1, 1)$, $s_2 = (-1, 1)$, $s_3 = (-1, -1)$, and $s_4 = (1, -1)$, then $E_{av} = 2$. In this special case, all symbols have the same energy.

Transmitter quality sets the base level quality of the signal that reaches the receiver. In the DOCSIS 3.0 standard, for example, QAM is used to transmit data as a series of symbols which are shaped with a pulse shaping filter. The filtering transforms the wideband energy pulses into a bandlimited signal for transmission. If a perfect, infinitely long pulse shaping filter were used, which is not implementable, it would not contribute to the intersymbol interference (ISI) in the received signal. However, since the pulse shaping filter used by the transmitter must have a finite length, some amount of ISI will be introduced by the transmitter. The collective impact of ISI and thermal noise is quantified by a measure referred to as the modulation error ratio (MER). This is the ratio of average signal energy to the mean squared error (MSE) in the decision variable. It is expressed in decibels (dB) as follows [11]:

$$\mathrm{MER}_{\mathrm{dB}} = 10\log_{10}\left(\frac{E_{av}}{\frac{1}{N}\sum_{j=1}^{N}|e_j|^2}\right), \tag{1.2}$$

where N is the number of samples being averaged, E_{av} represents the average symbol energy,

and e_j represents the error samples. E_{av} is dependent on both the modulation order and the symbol constellation.

In ideal operation, the topology of the cable network has no effect on the signal quality. Signal loss is corrected by filters and amplifiers along the cable distribution plant. However, the quality and age of the components that make up the network can significantly affect the signal. Deficiencies in the coaxial cable network such as wire faults, improperly terminated cables, or corroded connections cause impedance mismatches along the propagation path [14]. Part of the RF energy is reflected and re-reflected at these mismatched points, causing echoes in the signal. This effect, commonly known as multipath, is central to this thesis.

Consider an upstream transmission from a cable modem (CM) to the head-end cable modem termination system (CMTS). As the transmission travels through the cable, it is passed through a series of amplifiers and subscriber taps along the way [15]. If the connector to one of the amplifiers or taps is mismatched, it will reflect some energy back in the direction of the CM. This reflection will be re-reflected if it meets another impedance mismatch elsewhere. In such a case, it travels toward the CMTS and arrives later in the form of an echo. Echoes can themselves be reflected, and it is possible to have several re-reflected echoes. This scenario is illustrated in Figure 1.4 for the case of two echoes. The reflection amplitude at each mismatch is 25% of the incident amplitude.

The original signal meets the mismatch at d_1 and 75% is passed directly to the CMTS, arriving at time t_1. This is the main path which has no reflections and has travelled distance d_3. The remaining 25% reaches the mismatch at d_2, where 25% is re-reflected back towards the CMTS. Upon reaching the mismatch at d_1 for a second time, 75% of the re-reflection is passed to the CMTS after travelling a distance of $d_3 + 2\,(d_2 - d_1)$. It arrives at the CMTS at time t_2. The remaining re-reflection that reaches the mismatch is echoed in the same fashion to arrive at the CMTS at time t_3. Echo 2 has travelled a distance of $d_3 + 4\,(d_2 - d_1)$. It is clear from Figure 1.4 why cable plant echoes can be referred to as multipath. The signal arriving at the CMTS is the sum of the main signal and the echoes. Each component travels a different distance and therefore takes a different path to arrive at the CMTS.

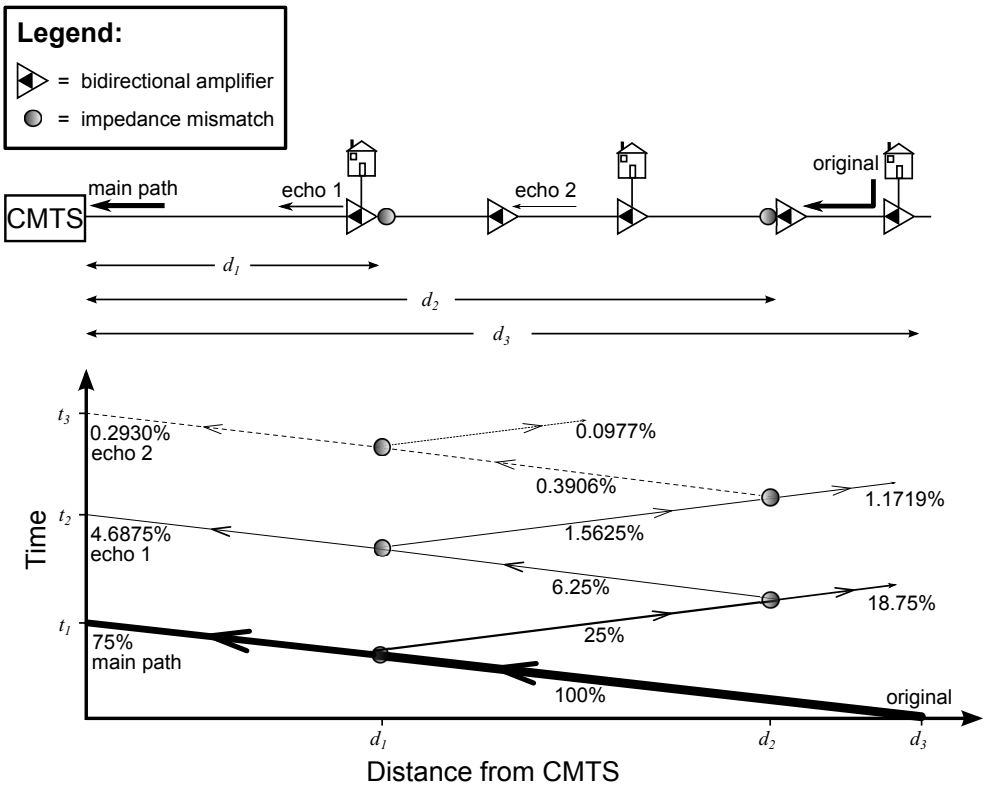


Figure 1.4: Upstream Cable Plant Echoes

Channel impairments such as cable plant echoes can be mitigated by the receiver through the use of an equalizer. Yet in order to do so the receiver must have some specific information about the impairments present on the channel. Before sending any data, the transmitter sends a pre-determined sequence of symbols often referred to as a preamble. This preamble sequence becomes distorted as it propagates through the channel and arrives at the receiver. However, the receiver knows what the correct sequence of symbols should be and can therefore compare the distorted sequence to the correct sequence to extract information about the channel. This information is then used to generate a set of complex correction coefficients (i.e. impulse response) to be used in the equalizer.

The ratio between an echo amplitude and the main path amplitude can be specified in decibels with respect to the carrier (dBc). These units are more meaningful than percentages, as echo strengths in DOCSIS are specified in dBc. The unit conversion is given by

$$\text{Echo Strength in dBc} = 20\log_{10}\left(\frac{A_{echo}}{A_{signal}}\right), \tag{1.3}$$

where A_{signal} represents the amplitude of the original signal, and A_{echo} represents the amplitude of the echo. If the signal amplitude percentages shown in Figure 1.4 are used with $A_{signal} = 75\%$ and $A_{echo} = 4.6875\%$, then 1.3 yields -24.08 dBc. Even low-level echoes can significantly increase ISI, and the digital equalization algorithms employed by CM receivers to mitigate the echoes tend to be costly. In order to test such receiver algorithms, the signal must either be sent over a physical multipath channel or something that emulates one. Not only is an emulation more economical than building a physical CATV network, but its parameters can be controlled precisely. Additionally, testing a different set of parameters does not require the construction of a different network topology.

1.1 Hardware Emulation Versus Software Simulation

Channel emulation can be performed by either hardware or software methods, or by using a combination of the two. A purely software driven approach would be the use of a simulation environment such as ModelSim to generate a testbench for the receiver. This approach allows testing of the receiver implementation without hardware synthesis. The simulation of the channel impairments could be performed inside the testbench and used as receiver input. However, depending on the size and complexity of both the FPGA and the design, this type of gate level simulation is extremely slow. Part of the reason for this is that the FPGA hardware is highly parallel, yet general purpose computer processors are very serialized in design. Every bit transition must be calculated along with estimated propagation delay. At best, each transition could consume a single computer processor cycle — in reality it is likely to be more. Even a relatively short 1 second simulation could take many hours or even days to finish, after which another simulation would be needed for a different set of channel parameters. Clearly this type of offline processing is very time consuming.

An alternative to a purely software based approach would be to combine the software and hardware approaches. A set of data can be generated and altered with a channel model constructed inside a program such as MathWorks Matrix Laboratory (MATLAB).

Figure 1.5: Channel Emulation With External RF Emulator

The output data can then be loaded into the device block memory for use as an input to the receiver that has been synthesized in hardware. Unfortunately, the amount of test data required quickly consumes the device memory and only very short simulations (less than 1 second) can be achieved. In order to effectively test a physical receiver, a real-time RF channel emulator is needed to emulate a CATV channel as it can operate continuously.

One common approach to receiver testing is to use an external RF channel emulator. This method has been discussed in [16] and [17]. It requires the use of a full transmitter which includes an analog back-end, and an analog front-end for the receiver. Furthermore, many of the commercial RF channel emulators that are available use state of the art components making them an inherently expensive piece of equipment. Although this approach (shown in Figure 1.5) provides a real-time emulation and is effective for the final stages of testing, it does not allow a fully digital receiver test.

1.1.1 Integrating the Emulator and the Modulator

Modern digital systems such as modems are often developed and implemented on field programmable gate arrays (FPGA). Combining flexibility and parallel processing with high clock rates, these devices are programmable microchips with a variable architecture. Using a hardware description language such as Verilog allows an entirely different system to be loaded into the device without the need to change the physical hardware. In essence, these devices provide system designers with the means to tailor the on-chip hardware to suit the needs of the application. This flexibility is invaluable during the early stages of hardware development as it allows test hardware to be included on the same device as the circuit being tested.

It is possible to rearrange the signal path in such a way that an RF channel emulation can be performed inside the modulator of the transmitter. By moving the physical location

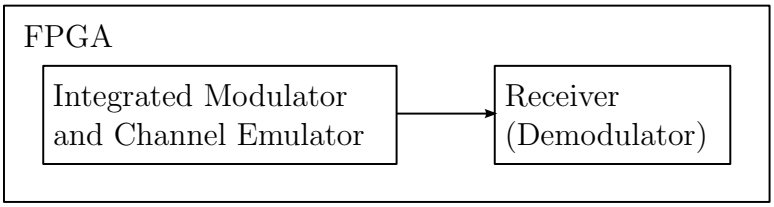


Figure 1.6: Channel Emulation Using Integrated Structure

of the emulator so that impairments are applied to the signal before upconversion to RF, a complex baseband equivalent channel is realized. This method is much less demanding in terms of hardware resources than the approach shown in Figure 1.5. Not only can this integrated modulator/emulator structure be included on the same FPGA as the receiver, it can be used in the absence of a receiver analog front-end to provide a fully digital real-time channel emulation at baseband. Although such an emulator must be designed and developed in addition to the receiver being tested, the associated cost savings often outweigh this inconvenience. This approach is illustrated in Figure 1.6.

1.2 Problem Statement

The central concern of this thesis is the implementation of an economical hardware circuit for DOCSIS receiver testing that is capable of replacing the need for a commercial RF emulator. Fully digital testing of different equalization, frequency, and timing recovery circuit implementations is the primary motive for creating the channel emulator. Frequency recovery and equalization are implemented with digital signal processing (DSP) algorithms that are most easily tested with a direct digital signal.

A stringent theoretical treatment of each building block in the emulator design is not the goal of this thesis, as the works of others are cited throughout. The aim is instead to accurately mimic a practical RF multipath channel in a digital emulator implemented on an FPGA. The focus is optimizing the hardware economy of the emulator. With this consideration, appropriate measures are taken to reduce the number of multipliers required by the components inside the emulator by taking advantage of the large number of registers available in the device. For example, successive approximation algorithms are used in the

place of hardware multipliers wherever suitable. Such an approach inevitably increases the register count and debug time of the system, but this is generally outweighed by the multiplier savings that are realized. Furthermore, development of a digital in-circuit emulator is a one-time cost in terms of development time and procurement. The emulators can be instantiated concurrently in multiple FPGAs at no additional per-unit cost.

This thesis proposes an alternative method for hardware receiver testing that is much less costly than conventional emulation. It proposes using the approach outlined in [18] to integrate the QAM modulator and channel emulator into a single structure, thus forming a complex baseband equivalent of the RF channel. Channel modelling is applied at complex baseband, thus eliminating the need for RF upconversion. The reduced baseband sampling rates allow a high level of multiplier time-sharing. The proposed architecture is also easily modified to suit the needs of different channel conditions (i.e. more signal paths, variation of parameters with time, different modulation schemes, etc.). Such an integrated modulator/emulator is an economical structure that is small enough (in terms of device resources) to reside in the same FPGA as the receiver under test, making it a cost-effective alternative to a commercial RF channel emulator.

1.3 Thesis Outline

Beginning with Chapter 2, the fundamentals of modelling a general communications channel are addressed. The notion of a complex baseband equivalent channel is developed as it is a means of obtaining an economical channel emulation. Deficiencies commonly encountered in CATV channels are then investigated one at a time and incorporated into the model.

In Chapter 3, the proposed hardware structure is presented in detail. This begins with an explanation of the conventions used in the schematic diagrams followed by a list of the various clock frequencies used in the system. Next, the overall system architecture is examined with a functional description of the system followed by a block diagram of the architecture. Key circuits that are central to the operation and efficiency of the system are presented by first discussing the motivation for their inclusion. Detailed schematics and functional descriptions

of these key circuits are then provided.

Chapter 4 is a vital part of this thesis, as the DOCSIS specification is used to determine the parameters and performance capabilities of the emulator. The relevant protocols and processing techniques used in DOCSIS-compliant systems are examined as are their effects on the MER of the received signal. Special emphasis is placed on the channel echoes and their effect on the signal quality. The MER to be achieved by the emulator is determined by establishing a noise budget for the system. Finally, the MER goal and DOCSIS spectral mask limits are used to determine the appropriate pulse shaping filter configurations to be used in the emulator.

Practical implementation considerations and emulator performance are presented in Chapter 5. Specifically, the compromises that were made in the design process are discussed, followed by an examination of the equipment and methods used for testing and verification of the system. The performance of a few critical circuits is scrutinized, followed by an examination of the best-case MER performance achieved by the implementation.

Finally, Chapter 6 sums up the document, recaps the results, and ends with some remarks regarding the flexibility of the proposed channel emulator structure to future enhancements. It is demonstrated that a complex baseband channel emulation can be feasibly implemented in the same hardware fabric as the receiver under test, expediting the testing of the receiver DSP circuitry.

2. Channel Modelling

A typical CATV network consists of many miles of coaxial cable, bidirectional amplifiers, subscriber taps, and CMTS equipment located at the head-end. A fully functioning CATV network cannot be used for testing as doing so would disrupt normal service. Alternatively, building a duplicate network solely for testing is too expensive and too difficult to reconfigure for a different set of channel conditions. For these reasons the CATV network, or channels therein, are either emulated in hardware or simulated in software according to a mathematical model of the channel.

A test platform that accurately models the channel is crucial for debugging and testing a communication system. The shortcomings of a system may not surface until a realistic channel is encountered. For instance, the adaptive equalization and automatic gain control circuits of a receiver cannot be adequately tested without using the kind of channel that would be typically encountered. The development of a realistic CATV channel model is the focus of this chapter.

2.1 Methodology

Before discussing RF channel emulation, it is important to understand that the information-bearing component of the RF signal can be expressed as a complex baseband signal. Besides mathematical convenience, this complex baseband equivalent is used in hardware. Rather than performing a DSP function at passband (i.e. at RF), the function can be implemented on the complex baseband equivalent signal. Performing the signal processing at baseband is much more efficient as it can be done at a lower sampling rate.

As addressed in [19], a passband signal can be represented by a complex low-pass band-

limited signal, $\mathbf{s}(t)$, that modulates a complex sinusoidal carrier $e^{j\omega_o t}$ with center frequency ω_o. The complex baseband equivalent is related to the RF signal by the following relation,

$$\mathbf{p}(t) = \Re\left\{\mathbf{s}(t)e^{j\omega_o t}\right\}, \tag{2.1}$$

where the symbol $\Re$ denotes the real part of the complex argument. As the desired RF signal $\mathbf{p}(t)$ is real, a well known property of complex numbers can be used to recast 2.1 in a different form. Given a complex number, $\mathbf{a}$,

$$\Re\{\mathbf{a}\} = \frac{1}{2}\left(\mathbf{a} + \mathbf{a}^*\right), \tag{2.2}$$

where $\mathbf{a}^*$ denotes the complex conjugate of $\mathbf{a}$. Applying this to 2.1 gives

$$\mathbf{p}(t) = \frac{1}{2}\mathbf{s}(t)e^{j\omega_o t} + \frac{1}{2}\mathbf{s}^*(t)e^{-j\omega_o t}. \tag{2.3}$$

This equation represents a real RF signal as the sum of 2 complex conjugate signals. This representation is a useful starting point for developing the channel model since it is this signal that becomes distorted during transmission.

2.2 Time Delay

Echoes due to multipath can be modelled using time delays, the implementation of which will be discussed in Chapter 3. Given a function of time, say $\mathbf{p}(t)$, a constant time delay of τ can be applied to the signal, which is represented by $\mathbf{p}(t-\tau)$. Time delay affects both the baseband (envelope) and passband (carrier) signal components of a complex baseband equivalent. A delayed RF signal can be expressed as

$$\mathbf{p}(t-\tau) = \frac{1}{2}\mathbf{s}(t-\tau)e^{-j\phi}e^{j\omega_o t} + \frac{1}{2}\mathbf{s}^*(t-\tau)e^{j\phi}e^{-j\omega_o t}, \tag{2.4}$$

where $\phi = \omega_o\tau$ is the constant carrier phase shift due to the time delay. Since the phase shift factors are also complex conjugates, 2.3 holds and the delayed RF signal is also real. It

is clear that the phase shifts can be decoupled from the carrier signal and included with the complex low-pass component of the signal.

As demonstrated in the next section, this representation allows multiple complex baseband paths to modulate a common RF carrier signal for a baseband channel emulation.

2.3 Multipath

The development of a multipath channel model is an extension of the concept of time delay presented in the previous section. In this thesis, multipath is viewed as a composite signal consisting of the sum of a main path and several echoes where the delay times and power in the echoes are with respect to the main path. Without loss of generality, the main path is assumed to have unity gain and no delay.

To better understand the scenario of interest suppose the main path, $\mathbf{p}(t)$, is combined with 3 echoes to form a multipath RF signal. The time delayed signals are attenuated and added to the main signal to form $\mathbf{r}(t)$, which has

$$\mathbf{r}(t) = \mathbf{p}(t) + A_1\mathbf{p}(t-\tau_1) + A_2\mathbf{p}(t-\tau_2) + A_3\mathbf{p}(t-\tau_3). \tag{2.5}$$

The echoes have gains A_n and delays τ_n. The effect of the time delay on each path is decoupled from the complex sinusoid allowing a sum of baseband paths to modulate a single complex carrier. For an RF signal exposed to $M+1$ paths, multipath is expressed as

$$\mathbf{r}(t) = \mathbf{p}(t) + \sum_{n=1}^{M} A_n\mathbf{p}(t-\tau_n). \tag{2.6}$$

At this point an important distinction must be made. The term multipath is often associated with wireless transmissions, where the multiple paths are physically separate [20]. Frequently such wireless multipath signals contain many signal paths with only a very small variation in the gain and delay — there is no dominant main path. This is in contrast to the upstream portion of a CATV network, where the multiple paths exist on the same physical medium and have a dominant main path component. The multipath phenomenon is referred

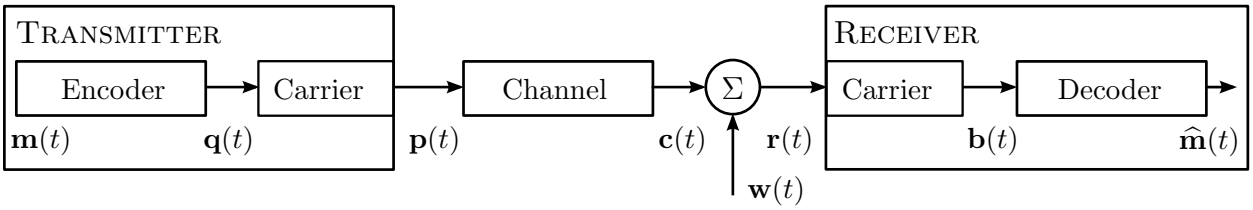
to as micro-reflections by the DOCSIS physical layer specification [11]. Going forward, any reference to multipath is intended to mean micro-reflections on a CATV network.

2.4 Thermal Noise

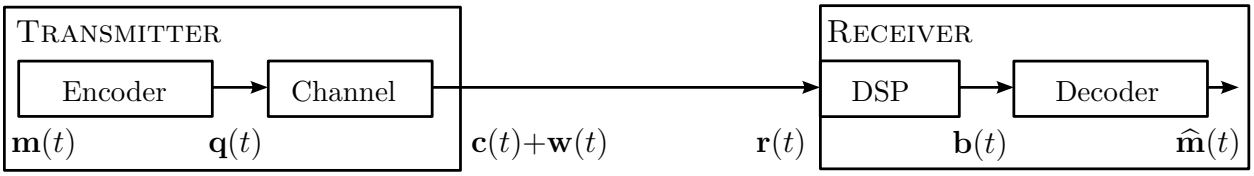
Thermal noise, which is an impairment common to all communication systems, is additive zero-mean noise. The spectrum of such noise is essentially flat, meaning the power in an interval depends only on the length of the interval and not the frequency at which it is located [21]. Noise with a flat spectrum is referred to as white noise, in reference to white light containing all frequencies in the visible spectrum. Thermal noise has an amplitude distribution that is Gaussian. Since it is also additive it is commonly referred to as additive white Gaussian noise (AWGN).

A block diagram of a general passband communication system is shown in Figure 2.1a. The AWGN is added to the signal at the channel output. The transmitter and receiver blocks are depicted with carrier blocks that represent the up/down conversion to/from RF. Here $\mathbf{m}(t)$ is the message data, $\mathbf{c}(t)$ is the channel output, $\mathbf{w}(t)$ represents the AWGN signal, and $\mathbf{r}(t)$ is the signal that is received. The additional signals are as follows: $\mathbf{q}(t)$ is the encoded signal, $\mathbf{p}(t)$ is the RF signal, $\mathbf{b}(t)$ is the baseband signal, and $\widehat{\mathbf{m}}(t)$ is the decoded message signal, which is a corrupted and distorted version of what was sent by the transmitter.

From Figure 2.1a, a digital representation of thermal noise $\mathbf{w}(t)$ must be injected. Frequency conversion to digital RF requires a higher sampling rate than than the baseband signal. Thermal noise along with the other channel impairments can be emulated at complex baseband. This is advantageous as it requires less hardware primarily due to the lower sampling rate. This is shown in Figure 2.1b where the channel is shown inside the transmitter at baseband. Note that $\mathbf{p}(t)$ is no longer in the figure as all paths have been accounted for by the baseband equivalent channel and are implicitly included in $\mathbf{c}(t)$. The model of Figure 2.1b has no RF upconversion or downconversion so the input to the DSP portion of the receiver is a complex baseband signal.



(a) A Generalized Passband Communication System



(b) A Complex Baseband Equivalent System

Figure 2.1: Equivalent Structures

2.5 Carrier Frequency Offset

Communication systems are designed in a way that requires the receiver to synchronize to the carrier. As the transmitter and receiver are physically separate devices, they use independent crystal clock oscillators so the receiver cannot be synchronized to a carrier generated in the transmitter. The frequency offset caused by independent reference crystals must be found and corrected by the DSP portion of the receiver.

Frequency offset is caused by components located inside the transmitter and receiver circuits rather than by the channel itself. While offset is normally encountered, the situation is different if the transmitter and receiver are implemented on a single FPGA and are referenced to the same crystal oscillator. In this case the complex baseband equivalent channel must include a special circuit to simulate the frequency offset. When expressed in terms of $\mathbf{c}(t)$ and $\mathbf{w}(t)$ this has

$$\mathbf{r}(t) = \mathbf{c}(t)e^{j\Delta\omega_o t} + \mathbf{w}(t), \tag{2.7}$$

where $\Delta\omega_o$ is the intentional frequency offset. A complex multiplication is applied to the channel normally located at baseband (i.e. when $\Delta\omega_o = 0$) in order to induce the desired frequency offset.

2.6 Incorporating Adjacent Channels

The upstream spectrum of a CATV network has a limited bandwidth and contains many tightly packed channels with power gains that can vary significantly. The combination of signals with large power gains in close proximity creates interference referred to as adjacent channel interference (ACI), which hinders the receiver.

Receiver performance can be more thoroughly tested if multiple channels are included in the model since a well designed receiver filter should remove them. Only the channels closest to the channel of interest (COI) have a significant effect, so only these adjacent channels are included in the model. Suppose the received signal is a complex baseband signal containing the COI, noise, and adjacent channels on either side of the COI. This is expressed as

$$\mathbf{r}(t) = \mathbf{c}(t)e^{j\Delta\omega_o t} + \mathbf{c}_1(t)e^{j\omega_1 t} + \mathbf{c}_2(t)e^{j\omega_2 t} + \mathbf{w}(t), \tag{2.8}$$

where the complex baseband components of the adjacent channels and the COI are $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}$, respectively. These have been frequency shifted to center frequencies $\omega_1 < \Delta\omega_o < \omega_2$ and in general, $|\Delta\omega_o| \ll |\omega_n|$. This creates a stack of 3 channels at complex baseband.

Although some of the methodologies of this chapter can be used to develop a general channel model for any application, the particular model presented here only applies to a wired CATV channel. This model cannot be used for a wireless communication channel as different physical phenomena and processes are encountered. For example, wireless channels are subject to specific types of signal drops, referred to as fading, which are not accounted for by the model developed in this chapter.

Having laid the foundation for the channel model which will be used going forward, a more detailed view of the channel emulator implementation can be provided. The next chapter explores the architecture used to implement the channel model presented in this chapter. The DOCSIS 3.0 physical layer specification will be examined later in Chapter 4, as the upstream channel conditions defined by this document are used to define the operational parameters of the channel emulator.

3. System Architecture

This chapter begins with a description of the schematic conventions to be used in this thesis. An overview of the emulator operation is provided next as a basis for an examination of key circuits within the emulator. The key circuits are either central to the overall operation of the channel emulator or have been structured with hardware cost in mind. Although many other important circuits are required to enable correct operation of the emulator, discussion of these circuits is deferred to Appendix A.

Before discussing the hardware specifics it is necessary to establish the notational and labelling conventions that are used in schematic diagrams:

- Signals marked `clr` go to the 'clear' input of all registers and are not shown explicitly connected.
- Clock connections to circuits and registers are omitted where such connections are obvious from the context. Circuits with multiple clock inputs are indicated with numbers beside the clock caret to match the corresponding clock signal.
- Data inputs to registers are shown with arrow tips.

Wherever possible, commonly used schematic and digital logic gate symbols are used in the diagrams. A diagram of the symbolic conventions used in the schematics is shown in Figure 3.1. As depicted, all sequential modules have one or more clock carets to indicate clock ports. Combinational modules have no such carets.

As is shown in Figure 3.1, different signal formats are used in the schematics. Some are shown with a single number while others have two numbers separated by a period.

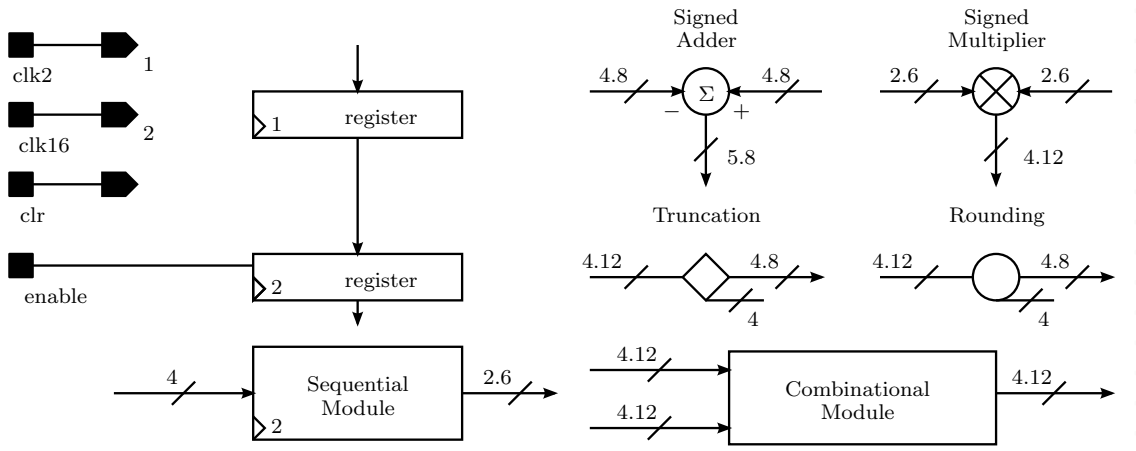


Figure 3.1: Conventions for Hardware Schematic Diagrams

The meanings of these signal formats are discussed here. First consider the single number format. Unless otherwise indicated in the schematics, a single number signal format is to be interpreted as an unsigned integer. For example, the signal format 4 refers to a signal that can have integer values from 0 to 15.

The interpretation is a little more complicated for the two number signal formats. Unless otherwise indicated in the schematics, these signals are to be interpreted as signed mixed format signals. That is to say, they are two's complement numbers with an integer component and a fractional component. The integer component of the signal is located to the left of the period and includes the sign bit. The fractional component is located to the right of the period. For example, consider a 2.6 signal format. This being a signed number, the most significant bit (MSB), which is the leftmost bit, has a value of -2. The next bit to the right has a value of $+1$. These are both of the integer bits in the signal. To the right of the period are the fractional bits. These have values $+2^{-1}$, $+2^{-2}$, $+2^{-3}$, $+2^{-4}$, $+2^{-5}$, and $+2^{-6}$ respectively. Thus this signal format can accommodate values from -2 to $2 - 2^{-6}$.

Truncation and rounding operations are denoted by diamond and circle symbols, respectively. Consider the truncation shown in Figure 3.1. The input signal format is 4.12, while there are two output formats: 4.8 and 4. Here the 4 least significant bits (LSB), which are fractional bits, are trimmed from the signal, leaving 8 fractional bits remaining in the

trimmed signal. The rounding shown here also has 4 fractional bits removed. Rounding of fixed point binary numbers is identical to rounding of decimal numbers. If the MSB of the bits being removed is a 1, then a 1 is added to the LSB of the remaining bits. Otherwise the remaining bits are unaltered. For example, suppose the 8-bit binary number 10010111 is rounded to 6-bits, with the 2 LSBs being removed. Since the MSB of 11 is 1, then $100101 + 000001 = 100110$ gives the result after rounding.

Clock name conventions are based on multiples of the symbol rate. For example, clocks `clk1`, `clk2`, `clk8`, and `clk16` operate operate at 1, 2, 8, and 16 samples/symbol, respectively. Clocks `n_clk8` and `n_clk16` are inverted with respect to their counterparts. A list of the clocks used by the top level of the emulator, their phases, and associated connection codes is given in Table 3.1.

Table 3.1: Clocks Used in Top Level of Channel Emulator

Clock Name	Frequency (MHz)	Phase (rad)	Connection Code
`clk1`	5	0	0
`clk2`	10	0	1
`clk8`	40	0	3
`n_clk8`	40	π	4
`clk16`	80	0	5
`n_clk16`	80	π	6
`clk32`	160	0	7

The on-board 100 MHz clock provides a reference for the phase-locked loop (PLL) circuit which generates the system clock signals for the emulator. This circuit instantiates the Altera IP generated by the ALTPLL megafunction.

3.1 Top Level Emulator Architecture

The top level of the hardware structure is presented first since it provides a good overall picture of the emulator operation. As is shown in Figure 3.2, the emulator architecture and operation are quite complex and require substantial explanation. The function of each major component shown in the figure is briefly described in the next subsection, and Figure 3.2 is referenced frequently.

3.1.1 Operational Overview

Loading the Parameters

Upon power-up/reset, which is initiated by pressing `button_0`, one of 4 user-programmable channel profiles is loaded from memory. There are 31 memory slots for channel parameters and loading these takes 2 symbol periods. The rest of the circuit does not begin operation until these 2 symbol periods have elapsed. Registers in the load parameters circuit (located at the top of Figure 3.2) hold the values cycled in from memory and provide them as inputs to other circuits. Any time the user changes parameters in the selected memory or selects a different channel profile with switches `SW[1:0]`, the registers are updated within two symbol periods.

Main Channel Transmit Data

Once channel parameters are loaded from memory, the main channel transmit data circuit (located in the upper left corner of Figure 3.2) takes over and begins reciting the preamble symbol data for the chosen main channel modulation mode. This preamble can be up to 256 symbols long and is user-programmable. A unique preamble can be programmed for each modulation scheme, with the same preamble being used for both the low-power and high-power QPSK modes. Upon completion of the preamble, the pseudorandom data generator begins generating raw (unmodulated) symbol data with an approximately uniform distribution. At the same time, a 6-bits/word by 65536 word RAM is enabled to store the generated data for off-line MER analysis.

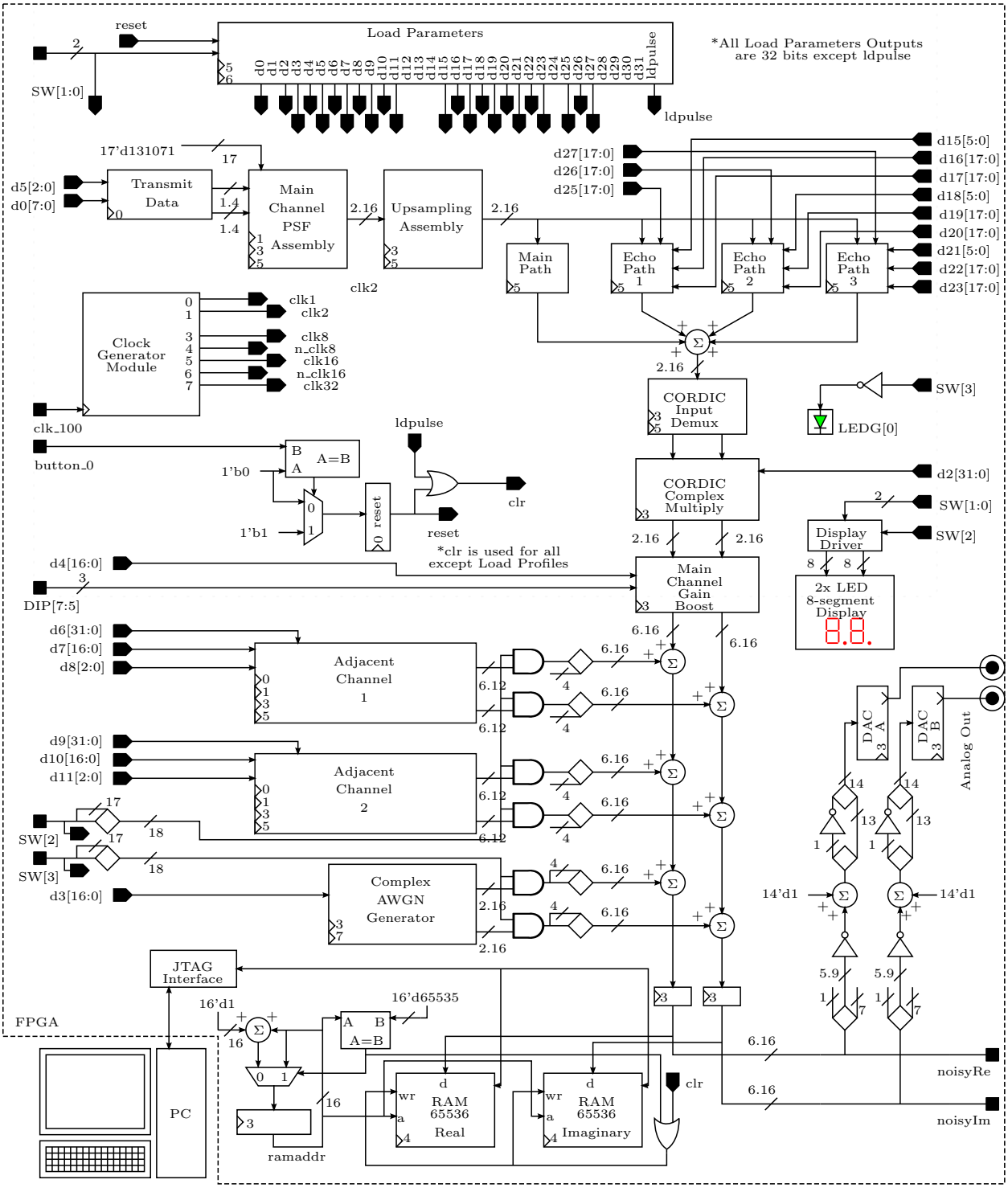


Figure 3.2: Schematic for Top Level of Channel Emulator

The preamble and generated data are mapped to constellation points by the symbol mapper. The symbol mapper encodes the raw symbol data into two streams (I and Q) of numbers in signed format with 1 integer bit and 4 fractional bits, which is referred to as 1.4 signed format. It is this complex stream that is used to modulate the I and Q carriers. Independent data must also be generated and mapped for each of the adjacent channels, but is much simpler than in the main channel since there is no need for a preamble. Furthermore, as these channels are not demodulated there is no need to store the generated data for analysis so no RAM module is included. This means the transmit data circuits for adjacent channels simply consist of a symbol data generator and a symbol mapping circuit.

Pulse Shaping the Symbol Stream

The mapped symbols proceed to the pulse shaping filter (to the right of the transmit data circuit) where the I and Q components are multiplexed into a single serial stream. This single structure performs the required spectral shaping on two independent streams of data, thus eliminating the need for duplicate parallel filter structures.

Upsampling by 2

The channel emulator must accommodate adjacent channels. This is done by generating 3 I-Q pairs of baseband signals at 4 samples/symbol and then upsampling each of these to 8 samples/symbol (i.e. 40 MHz clock) before frequency shifting. Doing so doubles the bandwidth available to the 3-channel stack. The output of upsampled I-Q pairs are interleaved (multiplexed) to make a single stream operating at a clock rate of 80 MHz. The upsampling assembly is located in the center of Figure 3.2, directly beneath the load parameters circuit.

Echo Paths

The upsampled output is then split into 4 paths: a main path and 3 echoes, located at the upper right corner of Figure 3.2. Each echo path delays and attenuates its signal. The main path simply applies a delay to compensate for the latency introduced by the echo paths — it applies no attenuation or variable delay. The echo paths are then summed to form a

multipath signal. The structures in the echo paths are also multiplexed to reduce hardware requirements.

CORDIC Input Demultiplexer

The CORDIC complex multiplier located beneath the echo path adder requires that the quadrature signal components be parallel rather than multiplexed. The CORDIC input demultiplexer takes the multiplexed I and Q samples as an input at a rate of 16 samples/symbol (80 MHz). The output is the parallel I and Q streams with a sample rate of 8 samples/symbol (40 MHz), making this module a serial to parallel converter.

CORDIC Complex Multiplication

Once demultiplexed, complex multiplication is performed on the baseband signal. Complex multiplication is commonly implemented using a ROM-based quadrature numerically controlled oscillator (NCO) and a pair of signed multipliers. An alternative approach is to implement a pipelined successive approximation circuit such as a CORDIC to realize the quadrature NCO. The CORDIC pipeline is initialized with the I and Q components of the signal being frequency shifted, thus eliminating the need for the hardware multipliers. Complex multiplication is performed on the adjacent channels as well as the main channel.

Main Channel Gain Boost

Located beneath the complex multiply circuit in Figure 3.2, the main channel gain boost is provided as a means of quickly increasing main channel gain without the need to edit the RAM. The default gain for the main channel is 0 dB. The DIP switches `DIP[7:5]` are provided for the user to implement main channel gains of +6, +12, or +18 dB if desired.

Channel Stacking

After the 3 signals have been frequency shifted to the appropriate center frequencies, they are summed to form a single 3-channel FDMA signal at complex baseband, as depicted by the vertical stack of quadrature adders on the right side of Figure 3.2. Adjacent channels

are toggled on or off by the user with `SW[2]`.

Complex AWGN Generator

The baseband channel stack is then added to complex AWGN from the thermal noise generator, forming the noise-laden 3-channel FDMA stack. This noisy signal constitutes the output of the channel emulator — the desired test signal. The generated noise floor spans the full 3-channel spectrum, with the specified SNR referenced from the main channel (i.e. the COI). Using main channel gain boost will therefore increase the SNR by 6, 12, and 18 dB respectively. The noise is toggled on or off by the user with `SW[3]`.

Output Signal RAM

Upon completion of power-up/reset and load parameters, the complex output RAMs (located at the bottom of Figure 3.2) are enabled and the first portion of the signal (i.e. the preamble and some data) is captured. The cumulative effects of all user settings (profile parameters, gain boost, and toggle switches) are accounted for in the captured signal. This RAM is provided for channel emulator performance measurements and is not necessary for a receiver testing scenario.

The noisy complex output signal is captured for 65536 samples at 8 samples/symbol. This allows analysis of a packet consisting of approximately 8000 symbols. Increasing the output storage capacity by using 8 pairs of page indexed RAM blocks would allow analysis of close to 65536 symbols, which is the number of symbols captured by the main channel transmit data circuit for MER analysis. The signal RAM word size is 22-bits for each of the real and imaginary components. Once the RAM is full, writing to RAM is disabled so that the preamble portion does not get overwritten, as it is useful in testing and verification. The captured data is saved to memory initialization files (MIF) on a PC and converted off-line[1] for analysis in MATLAB.

[1]Executable file `mif_convert.exe` written by Rory Gowen, University of Saskatchewan, 2013.

LED Indicators

A few of the different functions of the channel emulator are indicated with LEDs to make the emulator interface more user friendly. The most important of these indicators is the dual 8-segment LED display located on the upper left corner of the DE4 development board. This display shows which of the 4 channel profiles is selected by displaying `P1`, `P2`, `P3`, or `P4`. The decimal points are only lit when the adjacent channels are toggled on with `SW[2]`. For example, if channel profile 2 is selected (i.e. `SW[1:0]==2'b01` and adjacent channels are on (`SW[2]==1'b1`), then `P.2.` will be displayed.

There are 8 green LEDs located to the right of the dual 8-segment LED display. The right-most of these, labelled `LEDG[0]`, will be lit only when the AWGN is toggled on with `SW[3]`.

3.2 Key Circuits

3.2.1 Loading and Editing Channel Profiles

Background and Motivation

As with any user controlled equipment, the interface between the user and the hardware is very important. The DE4 development board contains a very powerful FPGA but, unfortunately, has relatively few switches and displays connected to the FPGA. This means the channel parameters cannot be entered through these switches. This necessitates the use of a JTAG interface[2] to edit the channel parameters.

Switches `SW[1:0]` on the DE4 board are used to select one of 4 preset channel profiles from internal memory. The parameters in memory are edited through the JTAG interface which can be accessed through the Quartus II software. The interface is set up this way to allow the user to quickly select one of the 4 profiles preprogrammed into the memory.

[2]Joint Test Action Group (JTAG) interface is a circuit embedded in the FPGA which provides access to the contents of RAM modules instantiated in the FPGA. The RAM contents can be written/read to a PC through this interface.

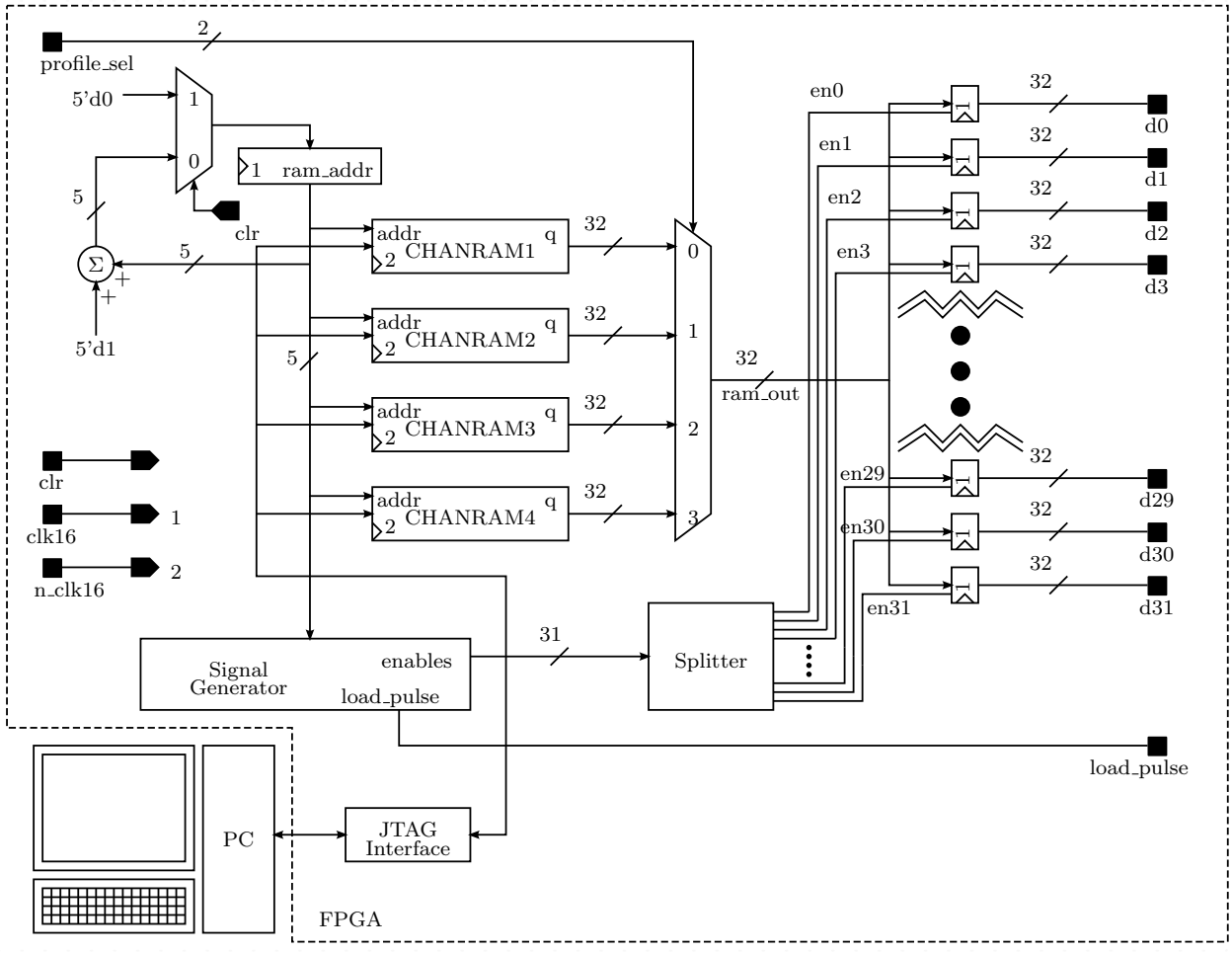


Figure 3.3: Schematic for Load Parameters Circuit

A circuit was designed that fetches channel parameters from memory and presents them to the various emulator circuits. The circuit is, in essence, a serial to parallel converter since it takes consecutive values made only briefly available by the RAM and turns them into parallel outputs that are held in a set of parameter registers. While transferring memory contents to the parameter registers after a power-up/reset, a pulse is generated to reset the channel emulator. This ensures all circuits have received the parameters needed for correct operation. The load parameters circuit is shown in Figure 3.3.

At the heart of the load parameters circuit are 4 RAM blocks. These RAM are 32-bits/word by 32 words (1 kilobit) each as all parameters used in the circuit are at most 32-bits long. The RAM address cycles all 4 RAM blocks simultaneously, which feed a data

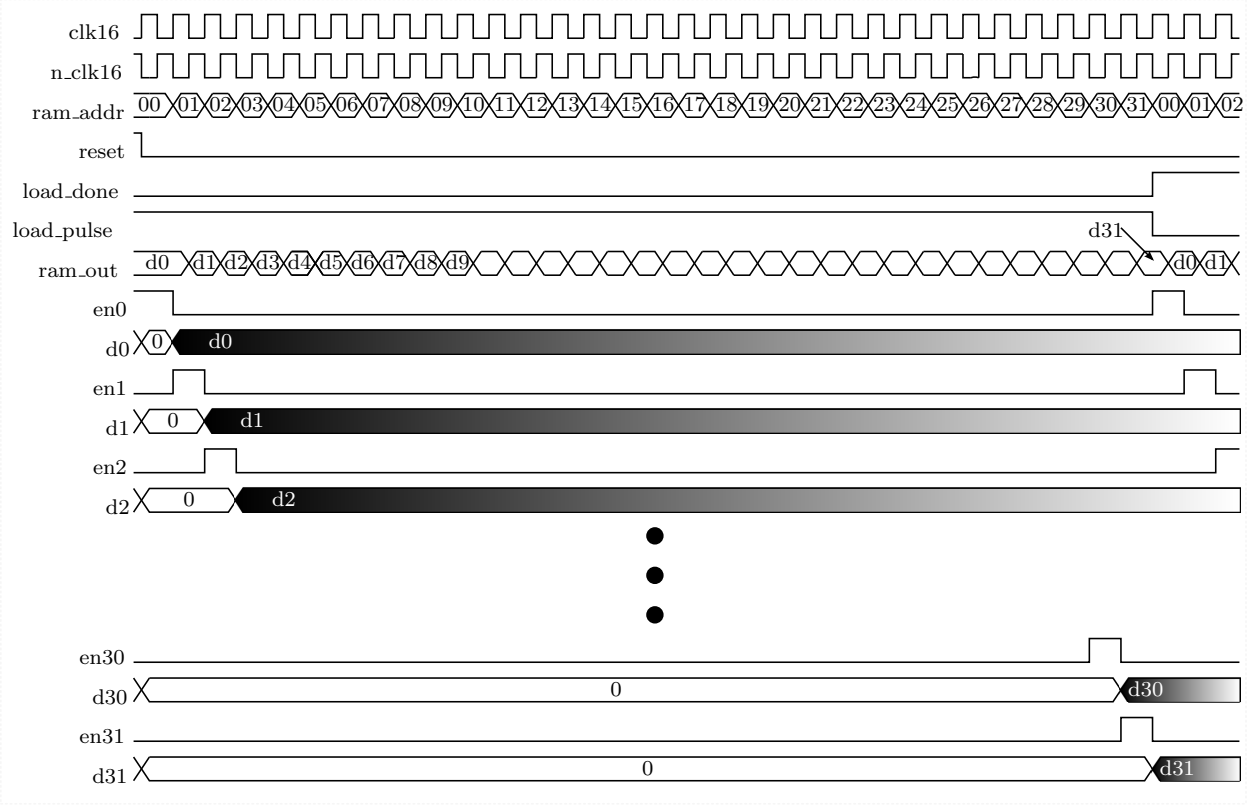


Figure 3.4: Timing Diagram for Load Profiles Signal Generator

selector controlled by the user through switches `SW[1:0]`. Only the RAM selected by the user updates the parameter registers, which hold the parameters until the next update (enable).

Generating Load Signals

Correct load signal timing is critical for the load profiles circuit. Output registers are enabled one-by-one as the RAM cycles through its contents. A timing diagram for the required load signals is provided in Figure 3.4.

The RAM data and address registers are written on the positive edges of `clk16` and the transfer to RAM is done with the negative edge of `clk16`. This is done so that there is only a single clock edge of delay between the address and the parameter register output.

Once the load parameters circuit detects that the final RAM address has been reached (i.e. `ram_addr==5'd31`), it sets the load pulse low and leaves it there until another power-up/reset command is encountered. This means that either the individual RAM parameters

or the entire channel profile can be changed by the user while the emulator is in operation.

Editing Channel Profiles

When the FPGA is configured with the channel emulator, memory on the FPGA can be written and read through the JTAG interface using the In-System Memory Content Editor while the circuit is operational. This allows the channel profiles to be altered without reconfiguring the device. Unfortunately, the memory content editor needs a very specific data format making it difficult to use.

The memory content editor uses hexadecimal format, but parameters are more meaningful to a user when specified in signed decimal format. For this reason, a MATLAB script was developed to convert signed decimal channel parameters into hexadecimal values to be used in the memory content editor and memory initialization files (MIF). This script, entitled `convert4rom.m`[3], converts user-defined channel parameters into equivalent 32-bit hexadecimal representations. Input values with units of Hz, ns, or dB, for example, are used to generate a list of unitless hex parameters to be entered into the memory by the user.

3.2.2 Main Channel Transmit Data Circuit

Background and Motivation

The transmit data circuit for the main channel is the starting point for the signal generated by the channel emulator. Its purpose is to recite the preamble symbol sequence, generate the pseudorandom symbol data, and map these symbols to locations in the complex plane.

Operational Overview

After the channel parameter registers have been loaded from memory, the main channel transmit data circuit begins operation. This circuit, shown in Figure 3.5, is always in one of two modes. In mode 1, which is the preamble mode, the circuit sequentially transfers the symbols held in the preamble memory to the symbol mapper. Once the circuit has

[3]Script file written by Andy Fontaine, University of Saskatchewan, 2013.

transferred the entire preamble it enters mode 2, which is the data mode. In this mode, the circuit generates pseudorandom symbol data to represent a data packet. The circuit stays in data mode until the next power-up/reset.

The main channel transmit data circuit houses the symbol generator and the symbol mapper, which are fully explained in Appendix A. Additionally, it contains 5 256-word RAM blocks, each with a different output word length. These 2, 3, 4, 5, and 6-bits/word RAM blocks hold preamble symbol data for modulation schemes QPSK, 8-QAM, 16-QAM, 32-QAM, and 64-QAM, respectively. Note that although there are 5 modulation schemes (i.e. quadrature modulation orders), there are 6 modulation modes as two QPSK modulation modes (low-power and high-power) are available. These are denoted by QPSK0 and QPSK1, respectively.

A preamble detector sets a flag when the preamble is complete and the circuit switches to mode 2, remaining there until the next power-up/reset. The main channel transmit circuit is shown in Figure 3.5.

3.2.3 Multiplexed Polyphase Pulse Shaping Filter

Nomenclature

There are 4 major sub-assemblies located inside the pulse shaping filter shown in Figure 3.6. Each is outlined with a dashed box. As they will be frequently mentioned in the subsections that follow, they are defined and explained here to reduce confusion.

- Data Pipeline: A general FIR filter structure consists of multipliers and adders which operate on a series of input data samples. The data samples are transferred sequentially through the filter using tapped registers in series (i.e. a shift register). The shift register that moves input data through the filter to be processed is referred to as the data pipeline.
- 4 MAC Stack: There are 4 MAC circuits (explained in the next subsection) used in the pulse shaping filter for the main channel. However, the pulse shaping filter used

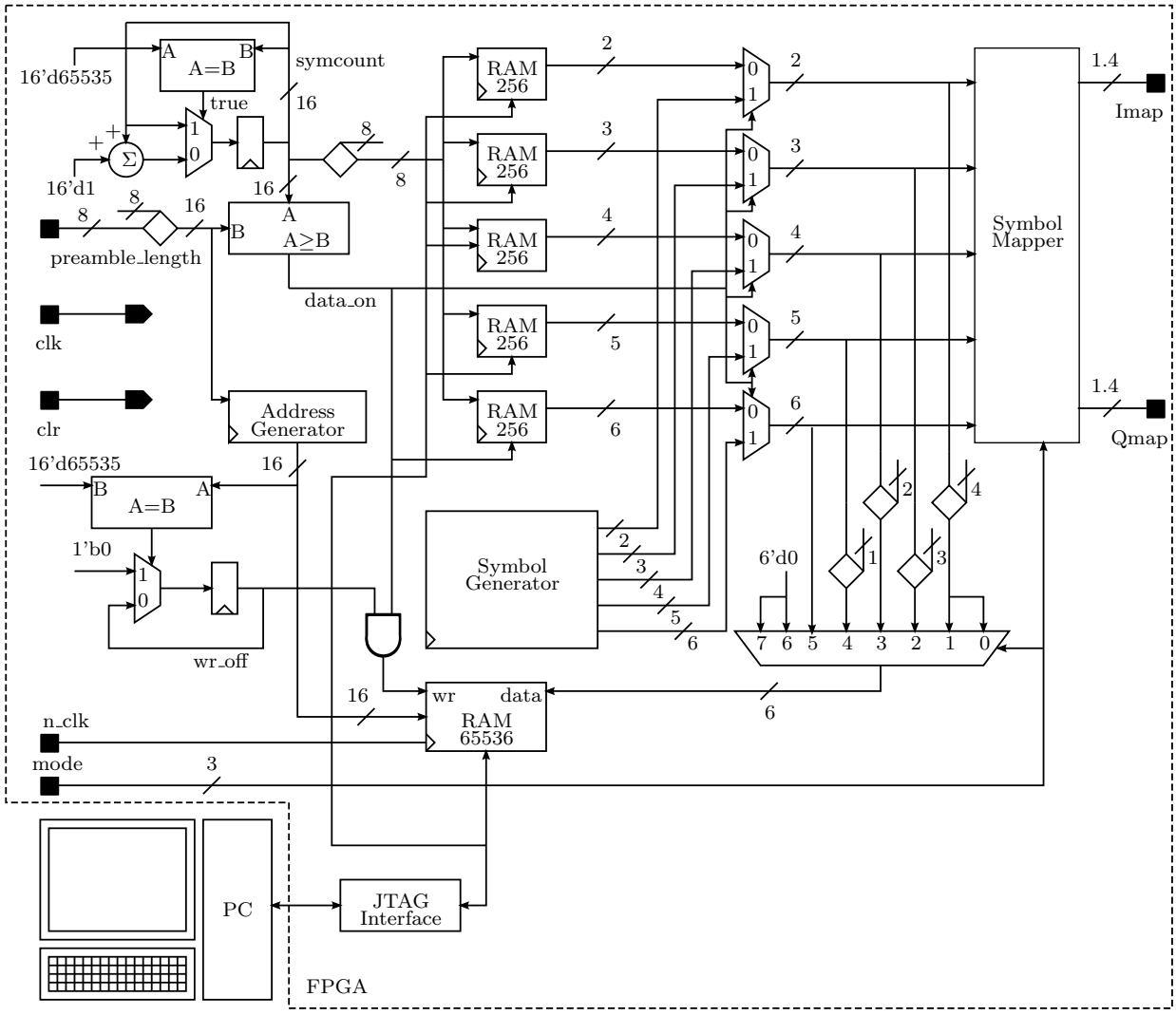


Figure 3.5: Schematic for Main Channel Transmit Data Circuit

for the adjacent channels requires 8 MAC circuits. To simplify the circuit schematic for the adjacent channel pulse shaping filter, the 4 MAC stack is shown as a module inside the adjacent channel pulse shaping filter described in Appendix A.

- Feedback Registers: This is a shift register used to transfer the filter output samples back to the multiplier to implement a signal gain adjustment.

- Feed Forward Registers: This is a shift register used to transfer the attenuated filter output samples forward to the output.

Background and Motivation

The input data (i.e. mapped symbols) must be upsampled before being transferred to the square-root raised cosine (SRRC) pulse shaping filter. This is crucial as the bandwidth of the pulse shaping filter exceeds the Nyquist frequency when the sampling rate, F_{samp}, is equal to the symbol rate, F_s. Normally the data is upsampled by a factor of 4 to comfortably satisfy the Nyquist sampling theorem. Rather than upsampling by zero-stuffing the input and filtering with a direct-form FIR structure, the FIR structure is decomposed into 4 phases which reduces the hardware required by the filter. This structure is easily implemented with a standard circuit referred to as a multiply and accumulate (MAC) circuit which further reduces hardware used in the implementation. The MAC uses a multiplier, an adder, a multiplexer, and a register to implement a segment of the filter consisting of several coefficients.

The pulse shaping filter is identical for both components (I and Q) of the signal. This allows yet another level of hardware reduction as the filter can be multiplexed to accommodate both signal components in a single structure. To accomplish this, the number of registers in the data pipeline is doubled, as is the frequency of the clock which strobes them. The number of multipliers and adders remains unchanged. The multiplexed filter structure can only be realized if the MAC circuits are also modified, which will be discussed shortly.

Operational Overview

A parallel input stream is multiplexed and passed through the data pipeline operating at 2 samples/symbol. The output of every second register is tapped and passed to an 8-to-1 multiplexer that toggles through its inputs at a rate of 16 samples/symbol. The multiplexer output is zero-padded with 13 fractional least significant bits (LSB) to form an 18-bit signed fraction format.

The filter impulse response has a length of 33 coefficients. The required filter length is determined using techniques which will be discussed in Chapter 4. With the selected clock rates, one MAC can implement 8 coefficients stored in lookup tables (LUT). This enables 32 coefficients to be implemented with 4 MACs. This leaves an extra coefficient that must be multiplied outside of the 4 MAC circuits. Another multiplier is time-shared to implement the remaining coefficient.

The input to the filter has a sample rate of 1 sample/symbol. The input data is upsampled by a factor of 4 and the filter structure is multiplexed, giving a filter output rate of 8 samples/symbol. As such, every parallel I-Q input pair results in a sequence of 8 output samples — 4 I samples followed by 4 Q samples. This sequence must be interleaved (IQIQ) before further processing by other circuits.

Architecture

The pulse shaping filter is implemented with the structure depicted in Figure 3.6. Four MAC circuits (dsmac #1 through dsmac #4) are organized so that each MAC computes the output of one of the 4 phases in the filter. The exception is the first MAC, which implements 8 of the 9 coefficients in the first phase of the filter. The other 3 phases contain only 8 coefficients each. The result from the first MAC is added to the remaining product from the final coefficient and fed back through the feedback registers for gain adjustment. After gain adjustment the samples are transferred through the feed forward registers.

Note that the pattern of two registers between taps in the data pipeline is broken for the last segment at the top of Figure 3.6 (top-right corner), which has 3 registers between

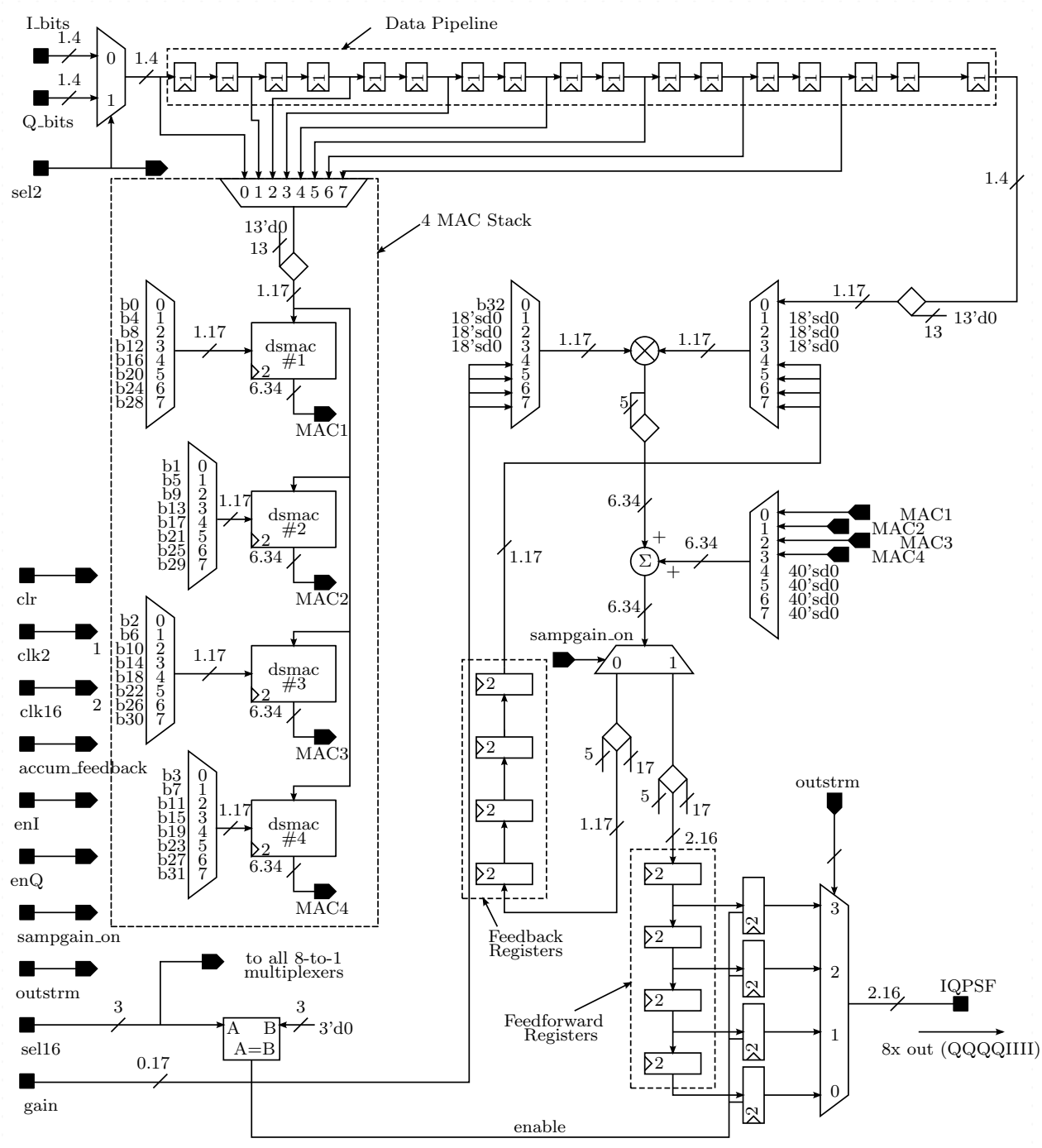


Figure 3.6: Schematic for 8 Symbol SRRC Pulse Shaping Filter

the taps. The third register imposes an additional delay of 1/2 symbol to the input of the multiplexer to wait until the outputs of the MACs are ready, which takes 8 `clk16` edges or 1/2 symbol period. This ensures that the final coefficient multiplication is correctly timed with the MAC1 output.

Dual Stream Multiply and Accumulate (DSMAC)

The pulse shaping filters used in this project, like most FIR filters, lend themselves well to implementation with MAC circuits as their output samples are composed of the sum of products. A typical MAC consists of a multiplier, an adder, a register, and a multiplexer which feeds the register output back to the adder. An initial product from the multiplier is loaded into the accumulator register on the first clock edge. For a specified number of subsequent clock edges, the multiplier products get added to the result being held in the register. When operated at a multiple of the filter sampling rate, MAC implementations result in significant hardware savings as compared to a direct form FIR structure as fewer registers, multipliers, and adders are needed.

As mentioned previously, further hardware savings can be realized if a single multiplexed structure is used to process the independent (I and Q) data streams. The MAC is customized to accept multiplexed input data at a rate of 16 samples/symbol for the pulse shaping filter used in the emulator. This reduces multiplier and adder requirements by a factor of one half.

The dual-stream MAC produces a multiplexed I-Q output at a rate of 2 samples/symbol, with each output sample (I or Q) being the sum of 8 products. The MAC accumulates 8 I products in `Iaccum`, then transfers the result to `Iout` which has now been enabled. At the same time, it begins accumulating 8 Q products in `Qaccum`, after which the result is transferred to `Qout` once it is enabled. The cycle then begins again. The schematic for the dual-stream MAC is shown in Figure 3.7.

Sequencer

A sequencer circuit is required to generate the correct timing signals to operate the MACs as well as the pulse shaping filter. This circuit consists primarily of sequential logic (registers

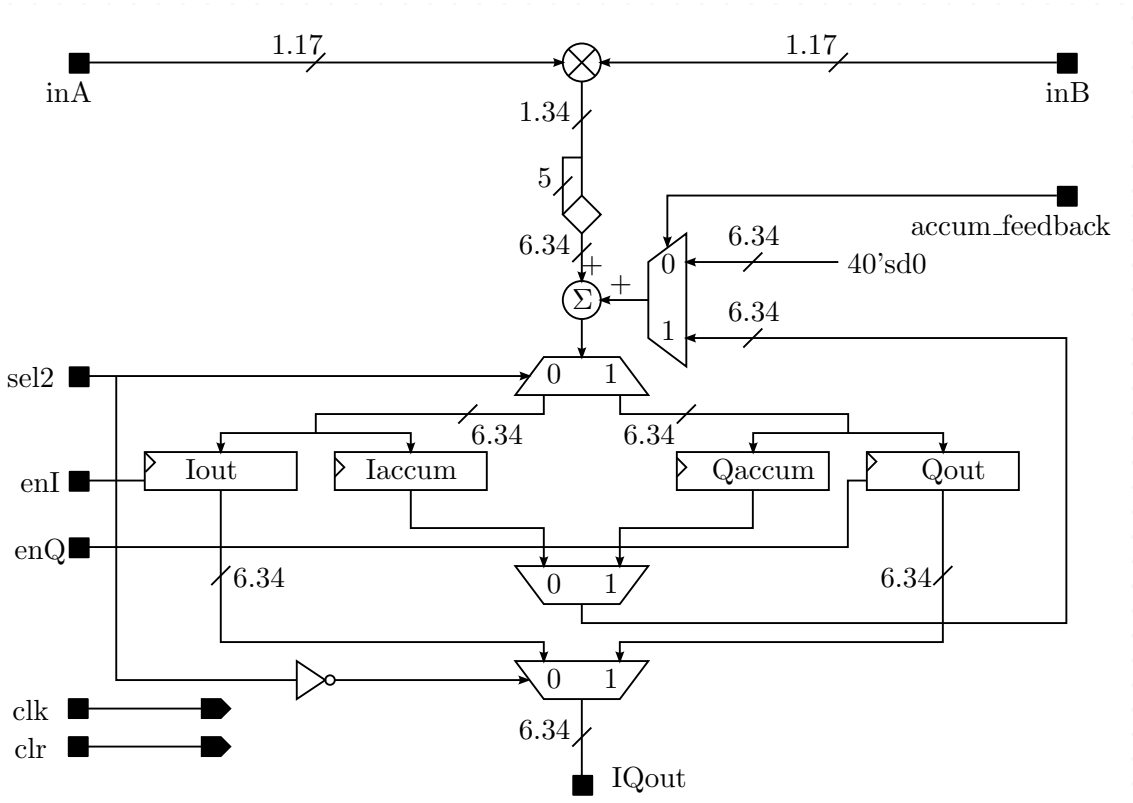


Figure 3.7: Schematic for Dual Stream Multiply and Accumulate (DSMAC)

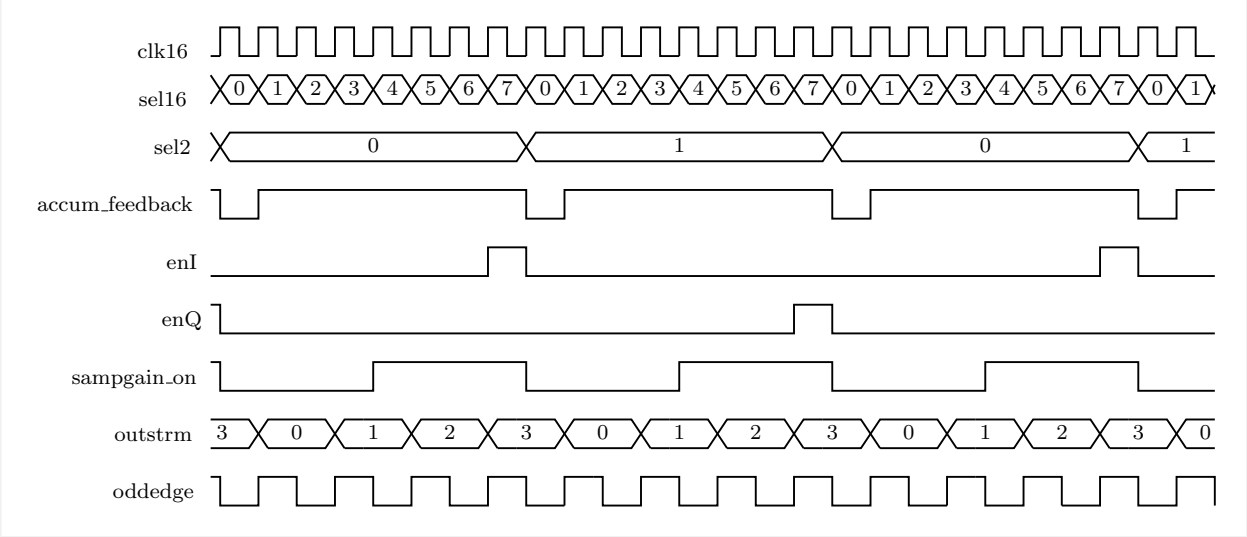


Figure 3.8: Timing Diagram for Dual Stream Multiply and Accumulate

and counters). A detailed circuit schematic of the sequencer is perhaps less informative than the timing diagram shown in Figure 3.8, which provides more insight to the behaviour of the circuit.

The I-Q enables are triggered on alternating occurrences of `3'd7` in `sel16`. The signal `sampgain_on` is low while the samples are being diverted to the multiplier, and high when the multiplier results are being sent to the feed forward registers on the output chain. Note that the signal `outstrm` is skewed from `sel16` by one `clk16` clock edge. This is due to the latency caused by the register between the the feed forward registers and the output multiplexer in Figure 3.6.

Main Channel Pulse Shaping Filter Assembly

The main channel pulse shaping filter assembly simply bundles the filter, sequencer, and interleaver into a single circuit. This is done solely to make the top level entity easier to manage - there are no other circuits in the assembly. This approach is also used for the adjacent channels. The schematic for the pulse shaping filter assembly is shown in Figure 3.9.

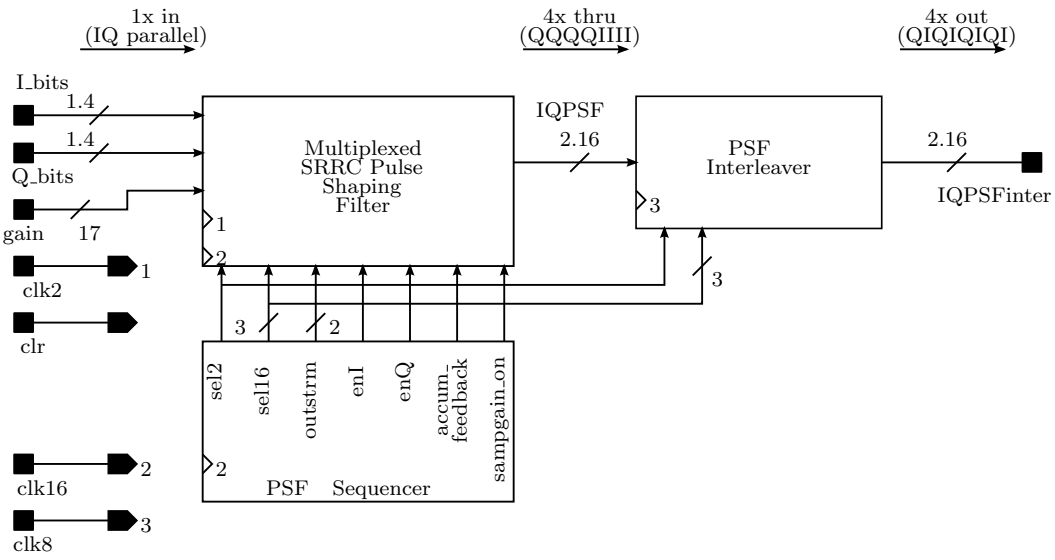


Figure 3.9: Main Channel Pulse Shaping Filter Assembly

3.2.4 Upsampling

Background and Motivation

Before discussing the need for upsampling after pulse shaping, the bandwidth of the pulse shaping filter output [22] is defined, which has

$$B = \frac{F_s}{2}(1+\alpha), \tag{3.1}$$

where F_s is the symbol rate in symbols/second and B is the bandwidth in Hz. The unitless parameter α is the rolloff factor of the SRRC pulse shaping filter, which for the application of interest is 0.25. This has a symbol rate of 5 Msymbols/second producing a bandwidth of $B = 3.125$ MHz, or equivalently, $B = 0.625F_s$. If this pulse shaped baseband signal is frequency translated to passband, the bandwidth is doubled so that $B_{pass} = 6.25$ MHz or $B_{pass} = 1.25F_s$.

A signal that contains 3 bit streams occupying a contiguous portion of the baseband spectrum can be constructed by summing the three bit streams, each of which occupies a channel. Doing so requires each of the single channel signals to be upsampled then fre-

quency translated to the appropriate region of the spectrum before forming the composite signal. One of these channels remains at baseband. Upsampling by a factor of 2, from 4 samples/symbol to 8 samples/symbol, is sufficient for the case at hand as total bandwidth occupied by the 3 channels is $B + 2B_{pass} = 1.875F_s$. This bandwidth is comfortably lower than the Nyquist frequency of $4F_s$. Without upsampling, the Nyquist frequency would be $2F_s$ which is not much higher than the bandwidth of the composite signal.

An alternative approach would be to use a pulse shaping filter operating at 8 samples/symbol, thereby achieving the desired sampling rate with a single hardware structure. Unfortunately, this method is inefficient as it doubles the number of multiplies needed for pulse shaping. Other structures exist which can perform upsampling using fewer multiplies.

Image Removal

Upsampling a signal is a two-step process. The first step is to zero-stuff the signal by a factor of L. To do so, $L - 1$ zero samples are inserted between the signal samples and the sample rate of the zero-stuffed signal is increased by a factor of L. The second step is to remove the spectral images in the pass band caused by zero-stuffing. This requires an image rejection filter, which is mathematically equivalent to an interpolator.

In order to determine the pass band and stop band edges for the image rejection filter, the bandwidth of the pulse shaping filter output must be determined using 3.1. The bandwidth is converted into normalized units through the relation $f = B/F_{samp}$, where f is the normalized frequency in cycles/sample and F_{samp} is the sampling rate of the pulse shaping filter in samples/second. For a pulse shaping filter running at 4 samples/symbol, the sampling rate is 20 Msamples/second, giving a normalized bandwidth of $f = 0.15625 = 5/32$ cycles/sample.

Zero-stuffing by $L = 2$ compresses the baseband signal along the normalized frequency axis so that its normalized bandwidth is halved. Since this signal must be retained, the normalized pass band of the upsampling filter must be at least $f_p = f/2 = 0.078125 = 5/64$ cycles/sample.

In addition to compressing the baseband signal, an image centered at 1/2 cycles/sample

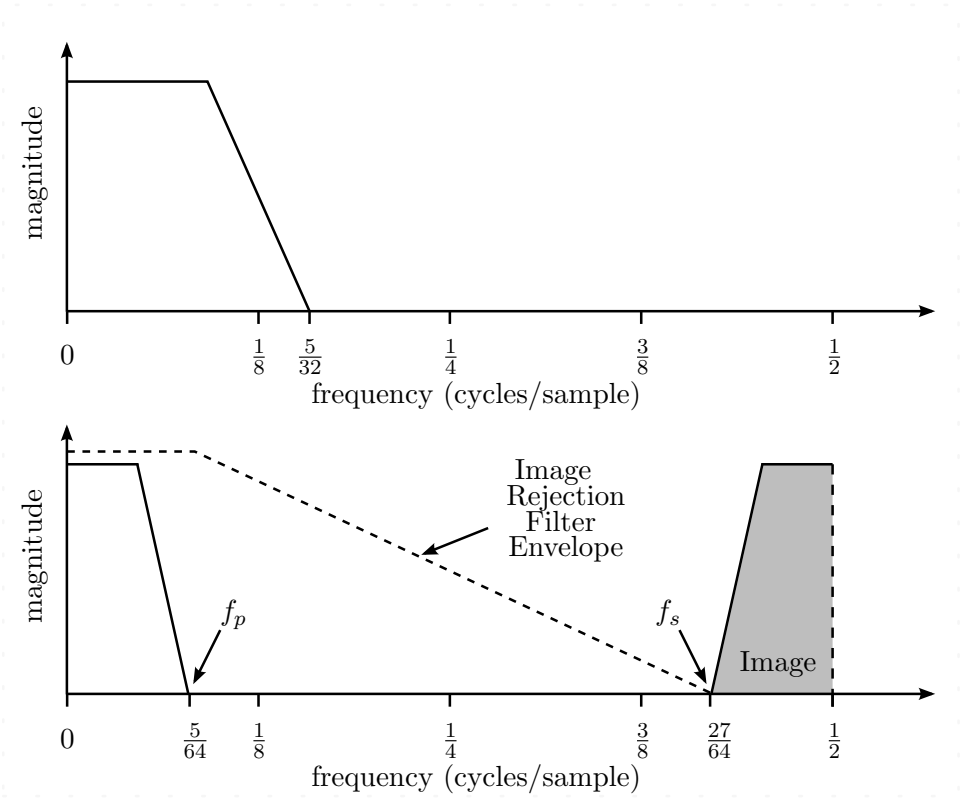


Figure 3.10: Determining the Stop Band Edge of Upsample by 2 Filter

is produced. The image has the same bandwidth as the compressed baseband signal and will span the pass band frequencies from $f_s = 1/2 - f_p$ to 1/2 cycles/sample. The scenario depicted in Figure 3.10 shows a stop band edge of $f_s = 0.421875 = 27/64$ cycles/sample.

Efficient Upsampling Structures

Finite impulse response (FIR) structures are used to implement filters with perfectly linear phase responses. Unfortunately when it comes to magnitude response, FIR filters do not perform as well as infinite impulse response (IIR) filters. FIR filters require more multiplications per sample to achieve a conventional magnitude response template specified with parameters of pass band ripple, δ_p, stop band ripple, δ_s, and transition bandwidth, $\Delta\omega$. The number of coefficients, N_{FIR}, needed for a classical equiripple FIR filter can be estimated by the Kaiser equation [23], which has

$$N_{\text{FIR}} = \frac{-10\log_{10}(\delta_p\delta_s) - 13}{2.324\Delta\omega} + 1, \tag{3.2}$$

where $\Delta\omega$, which is the normalized transition band, is expressed in radians/sample.

For example, a filter with a minimum stop band attenuation of 74 dB, a maximum pass band ripple of 0.01 dB, and a transition bandwidth of $\Delta\omega = \frac{11}{16}\pi$ radians/sample has parameters $\delta_s = 1.995 \times 10^{-4}$ and $\delta_p = 1.152 \times 10^{-3}$. Substituting these parameters in Equation 3.2 yields $N_{\text{FIR}} \approx 11.64 \approx 12$ coefficients.

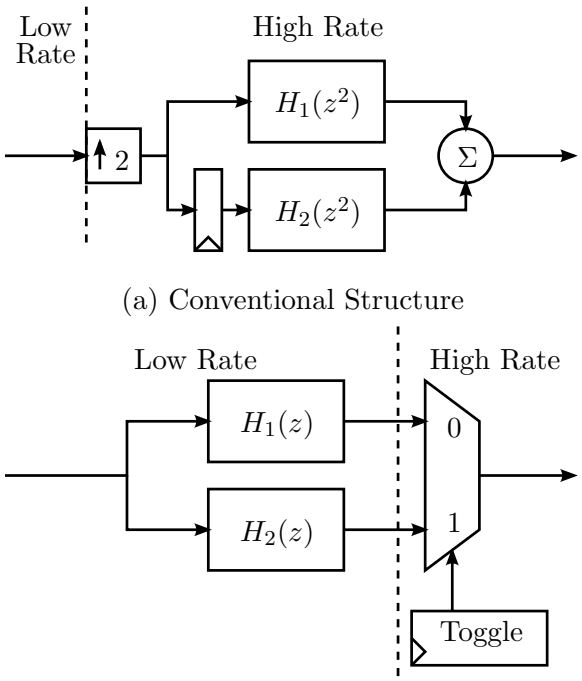
A filter with an input that is zero-stuffed for upsampling by a factor of 2 can be decomposed into two parallel filters as shown in Figure 3.11a. This structure is referred to as a polyphase filter. Due to the zero samples in the zero-stuffed input sequence, the outputs of the top and bottom filters are alternately zero. Thus, one of the inputs to the adder is always zero which is really a waste of an adder.

The polyphase structure of Figure 3.11a can be economically implemented by the structure of Figure 3.11b. Rather than explicitly zero-stuffing the input sequence and filtering at the output rate, which is twice the input rate, the sequence is directly filtered at the input rate. A commutator constructed from a data selector operates at twice the input rate to toggle between the two paths, generating an output at twice the input rate. Thus the adder can be replaced with a more economical commutator.

Operational Overview

The upsampling circuit resides in the upsampling assembly shown in Figure 3.2. The function of the upsampling circuit is to upsample the I and Q components of the signal by a factor of 2. Operationally, it separates the multiplexed input signals into two paths. Each path contains a different system function, the details of which will be examined shortly. An output commutator toggling at every `clk16` edge toggles the data selector between the two paths.

The upsampling filter has a multiplexed structure whose output is a stream consisting of two I samples followed by two Q samples, etc. As this ordering is incompatible with



(a) Conventional Structure

(b) Economical Structure

Figure 3.11: Equivalent Two-Path Filter Structures

the multiplexed structures encountered later in the processing chain, another interleaver is required to restore the I-Q multiplexed stream before further processing.

Architecture

The architecture of the upsampling circuit is shown in Figure 3.12. It is a low pass filter with a very nearly linear phase response that is implemented economically with a two-path structure, where one path consists of registers in series and the second path consists of an all pass filter (APF). Such structures produce half-band low pass filters that are well suited to rejecting the image created by zero-stuffing by a factor of 2. While they are economical, these structures do not exhibit a perfectly linear phase response. The phase distortion can be reduced to the point of being negligible at the expense of using a higher order APF in the second path.

There is an open-source MATLAB script[4] entitled `lineardesign_2.m` that computes the

[4]Script file `lineardesign_2.m` written by Dragan Vultec and fred harris of SDSU, 2002.

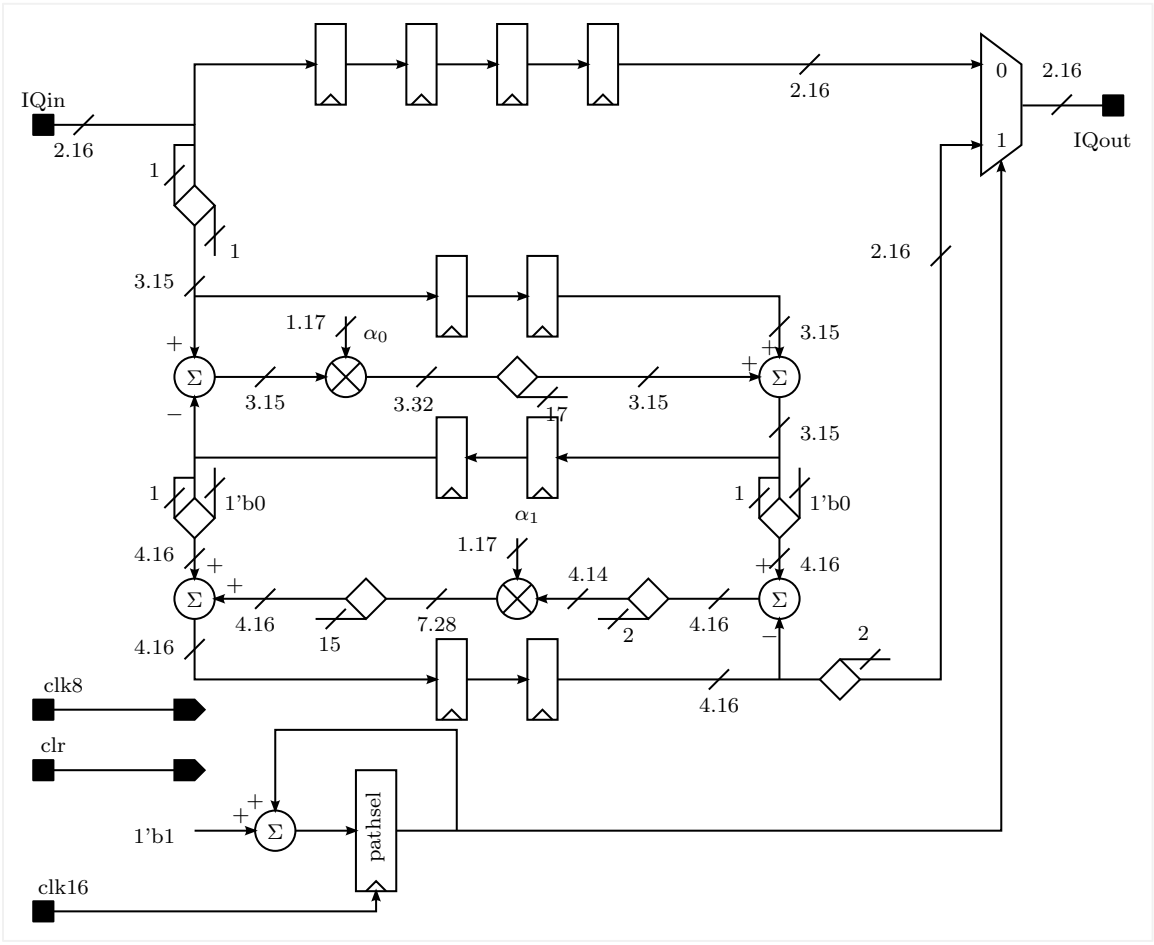


Figure 3.12: Schematic for Multiplexed 2-Path Nearly Linear Phase Upsampling Filter

delay on the first path as well as the APF structures and coefficients on the additional polyphase paths. This script is used to generate a two-path filter with two registers on the top path and two cascaded first-order APFs on the bottom path. A commutator operating at the output rate (i.e. twice the input rate) toggles between the two signal paths.

Setting up `lineardesign_2.m` so that it uses two paths, two coefficients, a stop band edge $f_s = 0.421875$ cycles/sample (obtained earlier), and 100 iterations, it specifies 4 registers in the top path corresponding to the system function $H_1(z^2)$. This means that implementing the economical structure requires two registers on the top path rather than 4. It specifies two cascaded first-order APFs on the bottom path with coefficients $\alpha_0 = 0.4854569008708127$ and $\alpha_1 = -0.07209196329829355$. These coefficients are rounded to 1.17 signed format for implementation. This filter achieves 74 dB of stop band attenuation while only requiring

two coefficients.

3.2.5 Multiplexed Pipelined Fractional Delay Filter

Background and Motivation

Delaying a signal by an integer number of samples is easily accomplished through the use of a shift register. However, delaying a signal by a fraction of a sample requires a more complicated circuit referred to as a fractional delay filter. Fractional delay filters are valuable signal processing structures whose theory, application, and implementation are thoroughly covered in [24] and [25]. A variable fractional delay filter, which is a fractional delay filter with an additional input that specifies the delay, allows each echo path to achieve virtually any amount of time delay. As cable plant echoes can have any delay time between certain limits, variable fractional delay filtering is needed to accurately mimic the micro-reflections encountered in the cable plant.

It can be shown[5] that the coefficients for a 3-tap direct form FIR fractional delay filter are $b_0 = \frac{\Delta^2-\Delta}{2}$, $b_1 = 1-\Delta^2$, and $b_2 = \frac{\Delta^2+\Delta}{2}$, where Δ is the fractional delay in samples from -0.5 to +0.5. In this case, the relationship between the 3 coefficients is such that only three multipliers are needed to compute 4 products: each of the 3 products of input samples and coefficients, and one to compute the square of the delay, Δ^2. Clever use of shifts and adds avoids unnecessary use of a multiplier, resulting in the structure[6] shown in Figure 3.13. This is a maximally flat[7] FIR filter with 3 coefficients.

As with the previous structures, multiplexing is used to reduce hardware requirements. The filter of Figure 3.13 must therefore have its registers and clock rate doubled to be compatible with the multiplexed I-Q signal. The resulting structure accommodates the

[5]This derivation was presented in a set of EE811 course notes on Practical Interpolation Filters provided by Dr. J. Eric Salt and Dr. Ha H. Nguyen, July 2012.

[6]Economical 3-Multiplier Fractional Delay Structure as presented in EE811 course notes on Practical Interpolation Filters provided by Dr. J. Eric Salt and Dr. Ha H. Nguyen, July 2012.

[7]A maximally flat filter has a magnitude response whose first and second derivatives with respect to ω are zero at the center of the pass band. For a low pass filter this occurs at $\omega = 0$.

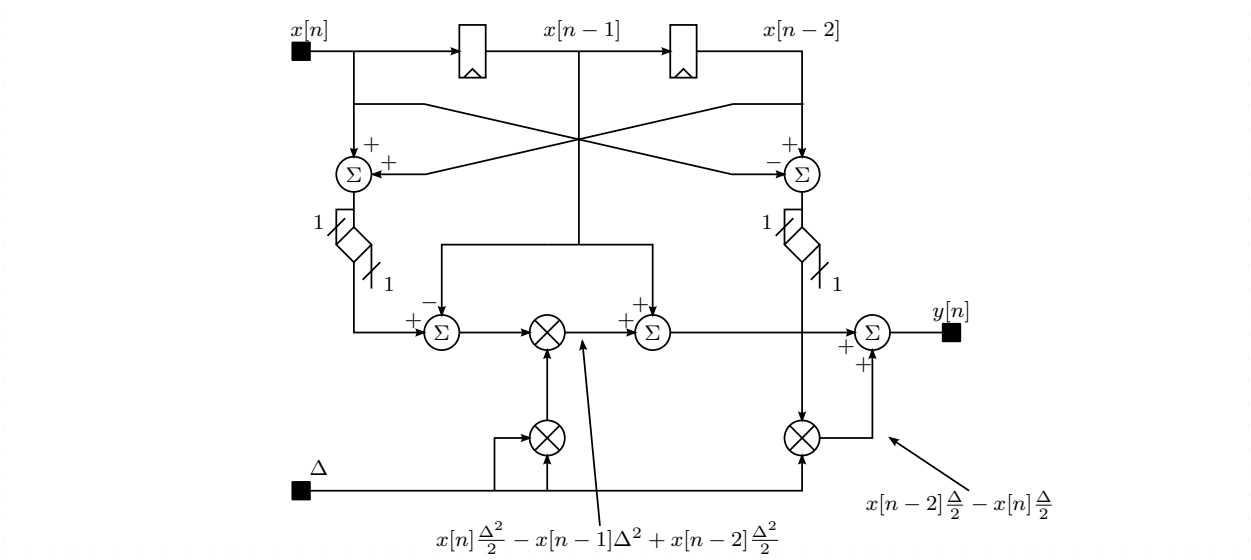


Figure 3.13: A 3-Tap Variable Fractional Delay Filter

multiplexed I-Q signals though it does have a weakness that is evident from Figure 3.13. The longest path from input to output has 5 consecutive combinational logic elements, which are 4 adders and a multiplier. As each of these imparts a propagation delay of a few nanoseconds, the cumulative propagation delay exceeds the clock period, which is $1/F_{\texttt{clk16}} = 12.5$ nanoseconds.

Architecture

The architecture of the fractional delay filter is shown in Figure 3.14. In order to decrease the register-to-register propagation time, additional registers are inserted and redistributed. These additional registers are known as pipeline registers. Careful implementation of this approach results in a circuit that can operate at a higher clock rate. However, the introduction of additional registers increases the latency[8] of the circuit. As can be seen in Figure 3.14, many registers have been added to the circuit shown in Figure 3.13. Any path taken from input to output has 6 additional registers so the latency of the pipelined circuit is increased by 6 clock periods. Since the sampling rate is 16 samples/symbol, the additional latency is 3/8 symbol. The latency of the original filter is 1/8 symbol, making the total latency

[8]Latency refers to the total delay from the input of a circuit to its output.

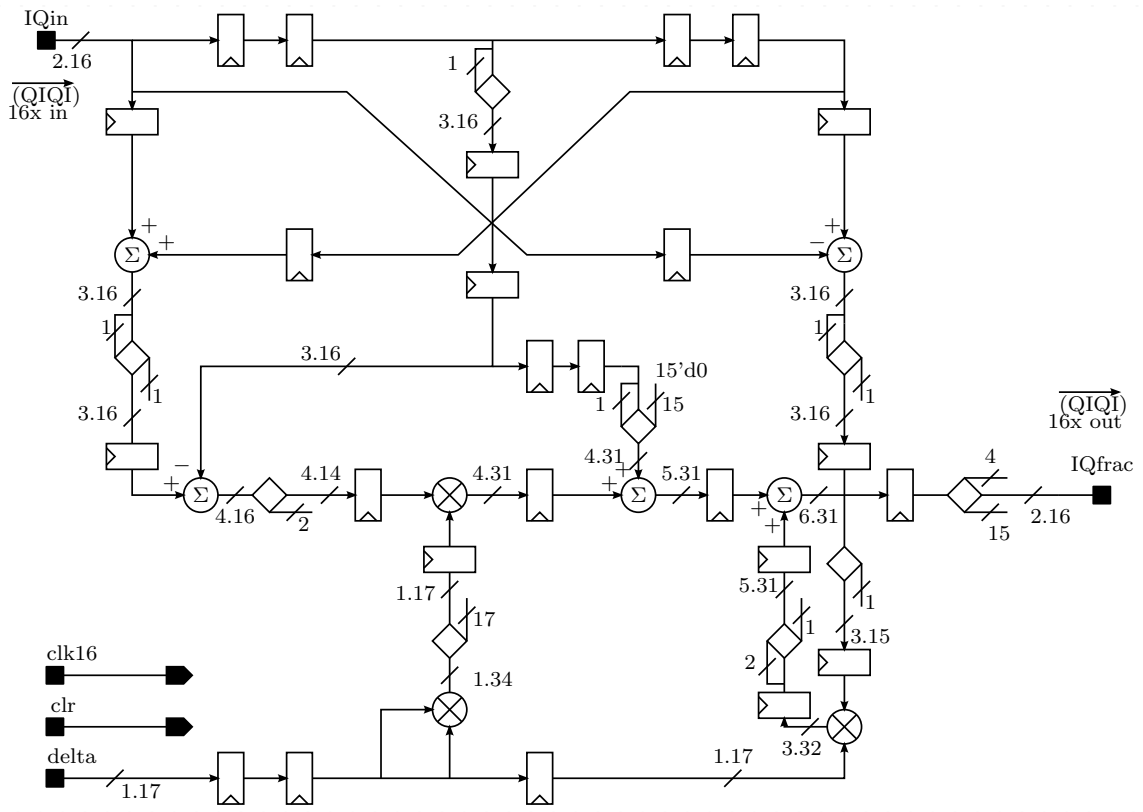


Figure 3.14: Pipelined Multiplexed 3-Tap Variable Fractional Delay Filter

1/2 symbol.

Unlike the filters that were previously introduced, the output rate of the fractional delay filter is the same as the input rate. There is no need to perform sample interleaving because this filter does not upsample — it is a first-in first-out (FIFO) circuit. The fractionally delayed output connects directly to the path attenuation module, which is presented in Appendix A.

3.2.6 Adjacent Channels

Background and Motivation

Adjacent channels are included in the emulator to test adjacent channel rejection of the receiver. For the purposes of verifying adjacent channel rejection, the adjacent channels must provide the same modulation schemes as the main channel but need not include any echoes.

However, for worst-case analysis, the power in the adjacent channels must be +20 dB relative to the main channel.

The adjacent channels do not need preambles so the symbol generator can start generating data immediately after the parameters have been loaded. This data must be independent for each of the 3 channels. As is shown in Appendix A.1, this is accomplished by assigning unique initialization seeds to each of linear feedback shift registers (LFSR) inside the 3 symbol data generating circuits.

Another notable difference in the signal processing chain for adjacent channels is the pulse shaping filter, which is 16 symbols long and uses a set of windowed impulse response coefficients. This filter therefore induces a longer delay than the 8 symbol filter used in the main channel. The internal timing of the MACs remains unchanged, but there are 8 MACs instead of 4. Additionally, gain is implemented with a combination of an internal multiplier and a bit shift (performed in the top level entity). Gain parameters are provided to the adjacent channel pulse shaping filters by the load parameters circuit and are not limited to 6 dB increments as in the main channel.

Operational Overview and Architecture

The architecture of the adjacent channels is shown in Figure 3.15. The operation of the adjacent channel circuits is straightforward. Pseudorandom symbol data is immediately mapped and transferred to the adjacent channel pulse shaping filter. After pulse shaping and interleaving the signal is transferred to the identical upsampling filter assembly used in the main channel. The multiplexed signals are then demultiplexed by a CORDIC input demux circuit similar to that used in the main channel. The parallel I-Q stream is then complex multiplied for frequency translation using a CORDIC-based circuit identical to that used in the main channel. The output rate is 8 samples/symbol.

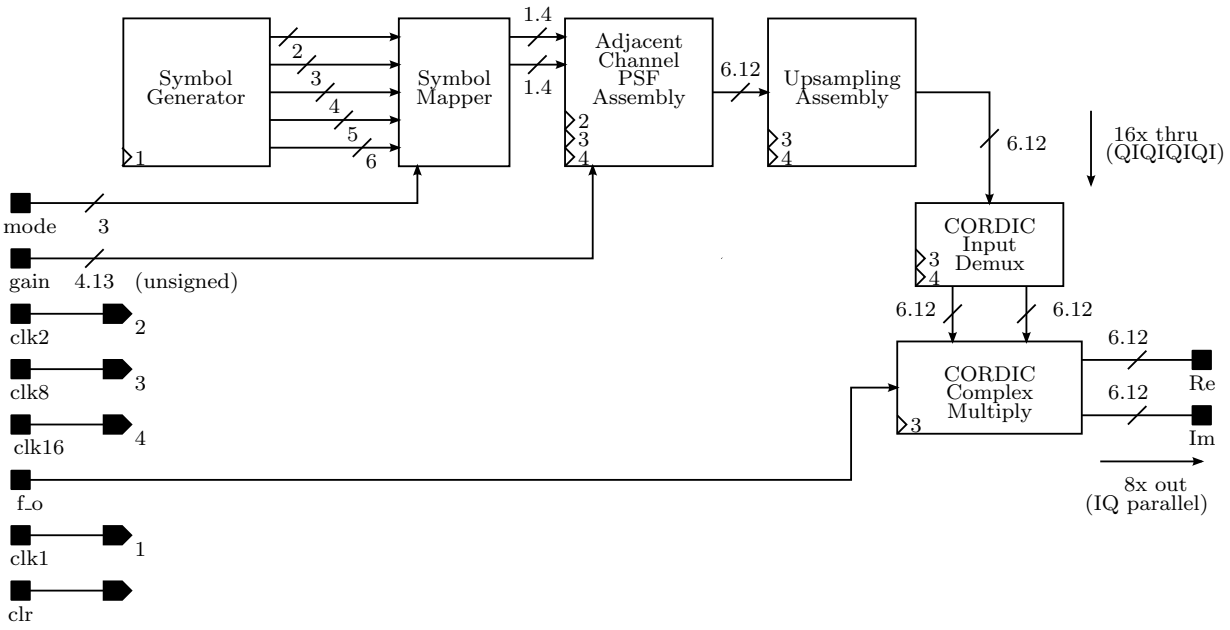


Figure 3.15: Schematic for Adjacent Channel Assembly

3.2.7 Complex AWGN Generator

Background and Motivation

Emulation of random thermal noise is necessary to create a realistic signal for the receiver. There are many ways this can be done, each with its own advantages and limitations. The method chosen for this project is not the most economical implementation, but provides high quality AWGN samples with very long pseudorandom sequences. This is advantageous for performing extended testing with a large number of very long data packets.

The method used to generate AWGN in the emulator is presented in [26] and combines an algorithm known as the Box-Muller method with the Central Limit Theorem to generate samples with an approximately Gaussian amplitude distribution. This method employs cosine, natural logarithm, and square-root functions and was chosen because successive approximation logarithm and CORDIC circuits had already been developed in previous work. The only other component remaining was the pipelined square-root circuit, and the decision was weighted more heavily toward a shortened development time rather than hardware economy. The schematics for the successive approximation circuits are presented in Ap-

pendix A.6.

The Box-Muller method takes two uniformly distributed random variables, x_1 and x_2, as inputs to create two functions of random variables: $f(x_1) = \sqrt{-\ln(x_1)}$ and $g(x_2) = \sqrt{2}\cos(2\pi x_2)$. Taking the product of the two functions returns a random variable with an approximately Gaussian distribution. That is, $G(x_1, x_2) = f(x_1)g(x_2)$.

The Central Limit Theorem states that as $N \to \infty$, the probability distribution function generated by averaging N identically distributed independent random variables approaches a Gaussian distribution [27]. In the implementation, only $N = 4$ consecutive noise samples are averaged. Since these samples are already approximately Gaussian, taking their average yields a better approximation with a minimal increase in hardware resources.

Operational Overview

Two identical parallel structures are used to generate complex uncorrelated AWGN samples. This overview describes the flow of the structure, which consists of two banks of LFSRs that are used to produce uniformly distributed 18-bit random variables at a rate of 160 MHz (i.e. using `clk32`). It uses a CORDIC to compute the cosine of one random variable, while the absolute value of the natural logarithm of the other random variable is used as an input to the square-root circuit to compute the second function. Once the functions of the two random variables have been computed, they are multiplied to produce a raw AWGN sample. Four such high rate samples (32 samples/symbol) are accumulated and averaged to form a single low rate AWGN sample at 8 samples/symbol. The noise level is set by an input from the load parameters circuit.

Architecture

The structure of the AWGN circuit is shown in in Figure 3.16. A parallel structure is chosen over a multiplexed structure for two reasons: 1) complex AWGN is added to the baseband components of the signal after they have been demultiplexed and, 2) the internal clock rate for the AWGN generator is 160 MHz (i.e. `clk32`) which is near the maximum clock rate for the circuit.

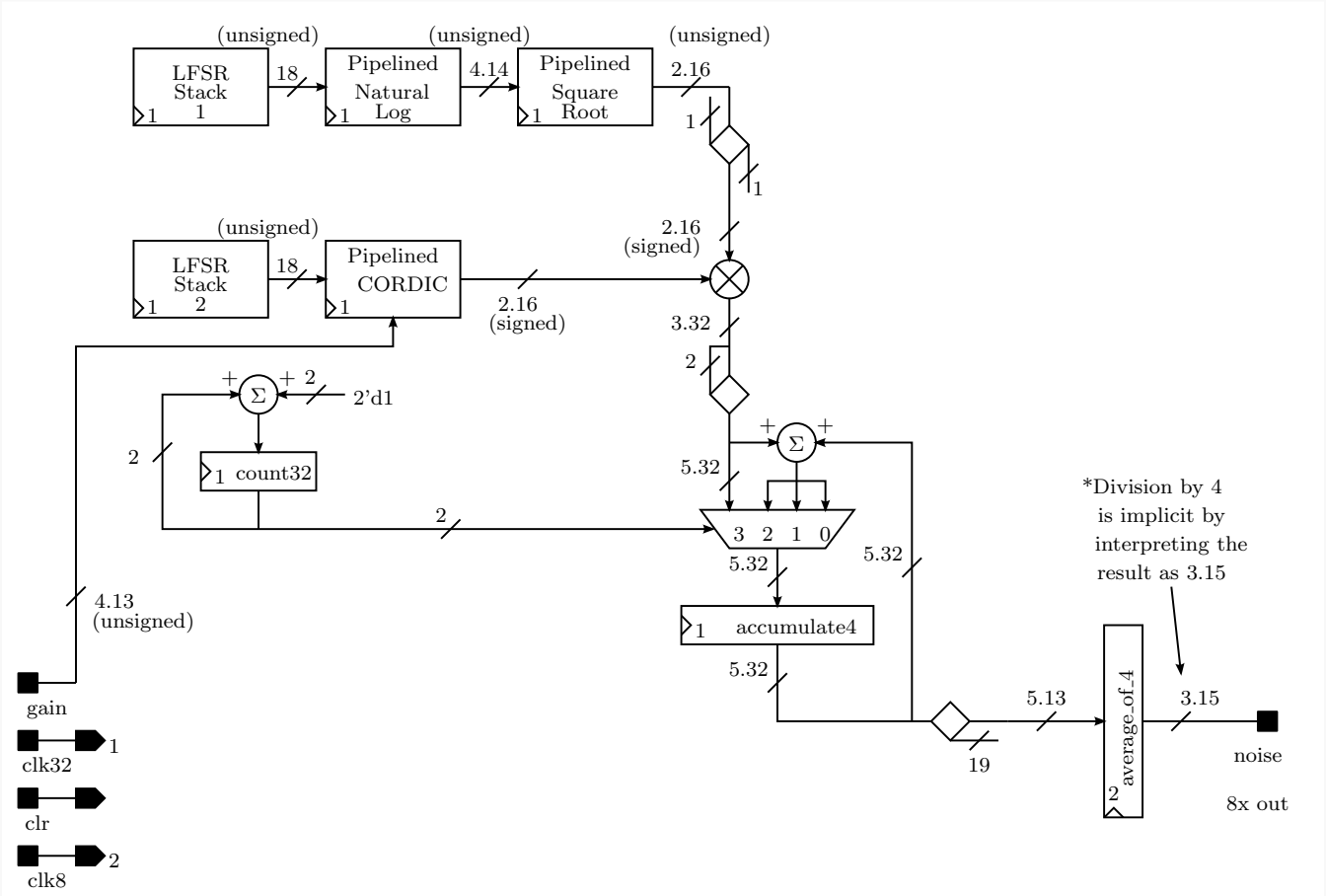


Figure 3.16: Schematic for Single Component (I or Q) of Complex AWGN Generator

The main shortcoming of the Box-Muller approach is its dependency on a multiplier to compute the product $G(x_1, x_2) = f(x_1)g(x_2)$. Implementing AWGN with a ROM-based approach could avoid the use of a multiplier at the expense of a large memory and some combinational control logic for addressing. However, the ROM-based approach is not investigated in this thesis.

4. The DOCSIS Specification and Channel Emulator Performance

In this chapter several details of the upstream channels defined by the DOCSIS 3.0 physical layer specification will be addressed. This specification, which is free and available for download from the CableLabs website, defines the protocols to be used in DOCSIS cable modems and networks between head-end and user. As these networks are implemented on existing CATV infrastructures and are subject to many different impairments, the specification outlines worst-case conditions for operation and the performance measures used to determine the quality of service.

The DOCSIS physical layer specification is an expansive document that addresses most aspects of the operation of a typical DOCSIS cable network. The vast majority of the information contained therein, although of interest to cable network operators, is beyond the scope of this thesis. The channel emulator that is the subject of this thesis is tailored to emulate a DOCSIS upstream channel. From this point forward, only characteristics specific to upstream transmission are discussed and elaborated in an effort to determine the desired channel emulator performance characteristics. The focus of this thesis is not to construct a complete cable modem (CM) for transmission, but rather to construct an emulator that will generate statistically similar signals to be used for receiver testing.

4.1 Mode of Operation

Before addressing the specific parameters to be implemented in the emulator it is necessary to define the operational environment. According to the DOCSIS specification, the physical layer employs CMs operating in either of two different formats: TDMA mode or

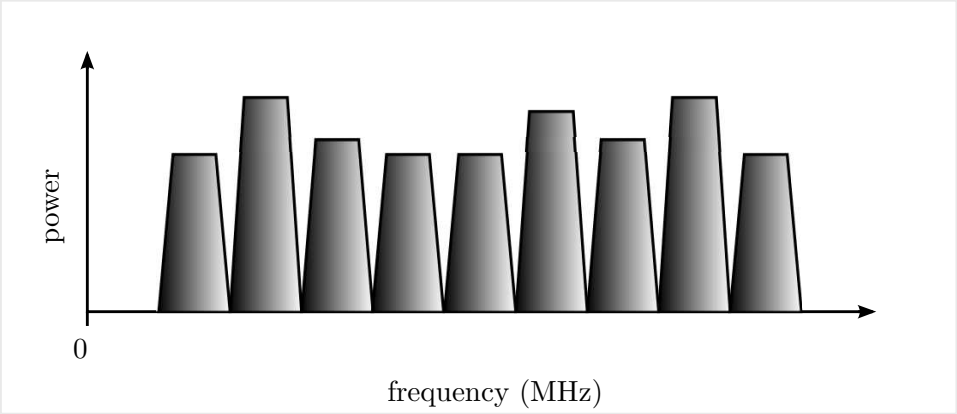


Figure 4.1: A Typical FDMA Spectrum

S-CDMA mode [11]. These operation modes allow for up to 6 different modulation rates (symbol rates, also known as baud). In TDMA mode, which will be explained shortly, there are 6 different modulation modes available for use. The operation mode, modulation rate, and modulation mode to be used by a particular CM is determined by the CMTS at the head-end.

Both the TDMA and S-CDMA modes operate in a format known as frequency division multiple access (FDMA). In this format, the available upstream spectrum contains numerous RF channels carrying different data signals in different frequency bands (i.e. at different center frequencies). In the ideal case of a 'brick-wall filter', these channels do not overlap or interfere with each other. However, since an ideal digital filter must have an infinitely long impulse response, brick-wall filters cannot be built and there is always some level of interference between channels. In addition, there may be noisy regions in the spectrum where the signal power of a particular channel may need to be increased in order to maintain a reasonable SNR, thereby exposing adjacent channels to increased interference. An example of an FDMA spectrum is shown in Figure 4.1.

In addition to FDMA, both operation modes also employ a technique known as time division multiple access (TDMA). Rather than transmitting data continuously, the CMs transmit their data in bursts. However, the CMs assigned to a particular channel must do so in an orderly fashion so that the CMTS at the head-end can make sense of all of the

data coming in. To achieve this, the MAC layer of the CMTS assigns time slots to the CMs on the network. In essence, the CMs transmit their data one at a time in their scheduled time slots so that the CMTS knows which data is arriving from which CM. Only the TDMA mode is implemented in the channel emulator. The S-CDMA mode is outside the scope of this thesis.

4.1.1 Modulation Rate

The stream of binary data to be transmitted is represented by a sequence of symbols mapped in the complex plane. Obviously, transmitting the symbols at a higher rate allows an increased data throughput. However, this increased throughput also comes at a price since the channel must span a larger frequency interval (i.e. higher bandwidth) to accommodate the data. When the upstream spectrum is crowded with many CMs transmitting to the head-end, it may not be possible for all of the CMs to transmit at the highest possible modulation rate.

The DOCSIS 3.0 specification allows for 3 modulation rates: 1280 kHz, 2560 kHz, and 5120 kHz. Earlier DOCSIS specifications (pre-3.0) used lower modulation rates of 160 kHz, 320 kHz, and 640 kHz. A manufacturer may choose to include all 6 modulation rates in their CM to allow for backwards compatibility with older CMTS equipment, but they are not required to do so. The different modulation rates do not change how a receiver detects and demodulates the incoming data. The same algorithms are employed regardless of which modulation rate is used. Therefore, in order to simplify the problem and reduce hardware requirements, only the highest modulation rate of 5120 kHz is used in the implementation of the emulator.

4.2 Modulation Modes

The upstream portion of a TDMA DOCSIS spectrum has 6 available modulation modes: QPSK0 (low power), QPSK1 (high power), 8-QAM, 16-QAM, 32-QAM, and 64-QAM. Both QPSK modes transmit 2 bits/symbol. The QAM modes transmit 3, 4, 5, and 6 bits/symbol respectively. Higher data throughput is achieved through the use of higher order modula-

tions, however the presence of significant channel impairments may cause too much signal degradation and lower order modulations must be used. The modulation mode to be used by a particular CM is determined by the CMTS at the head-end.

4.2.1 Modulation Constellations

The modulation constellations used for upstream transmission are given in Figure 4.2. The I-Q axis labels have been omitted for clarity. The average symbol energy, E_{av}, varies depending on the constellation. It is computed by summing the squared magnitudes of all symbol locations and dividing by the number of symbols in the constellation. That is,

$$E_{av} = \frac{\sum_{k=1}^{K} I_k^2 + Q_k^2}{K}, \tag{4.1}$$

where K is the number of symbols in the constellation, I_k is the coordinate of the real component of the k-th symbol, and Q_k is the coordinate of the imaginary component of the k-th symbol. The average symbol energy is necessary for computing the modulation error ratio (MER) introduced in Chapter 1, which is repeated here for convenience:

$$\text{MER}_{\text{dB}} = 10\log_{10}\left(\frac{E_{av}}{\frac{1}{N}\sum_{j=1}^{N}|e_j|^2}\right). \tag{4.2}$$

where e_j is the energy contained in the error vector and N is the number of symbols over which the MER is computed, which is generally much more than the number of symbols in the constellation. The average symbol energy for each constellation has been included in Figure 4.2. The power gain relative to the 64-QAM constellation, G_{rel}, is also included in units of dB.

The coordinates used in Figure 4.2 are used for mapping the data to distinct vector locations, (I,Q), in signal space. Since the values on both axes are located at integer values and have a maximum range from -14 to 14, then each of the I and Q components of the mapped symbols can be represented by a 5-bit signed integer. However, since the locations are all even values only a 4-bit signed integer is needed. The least significant bit (LSB)

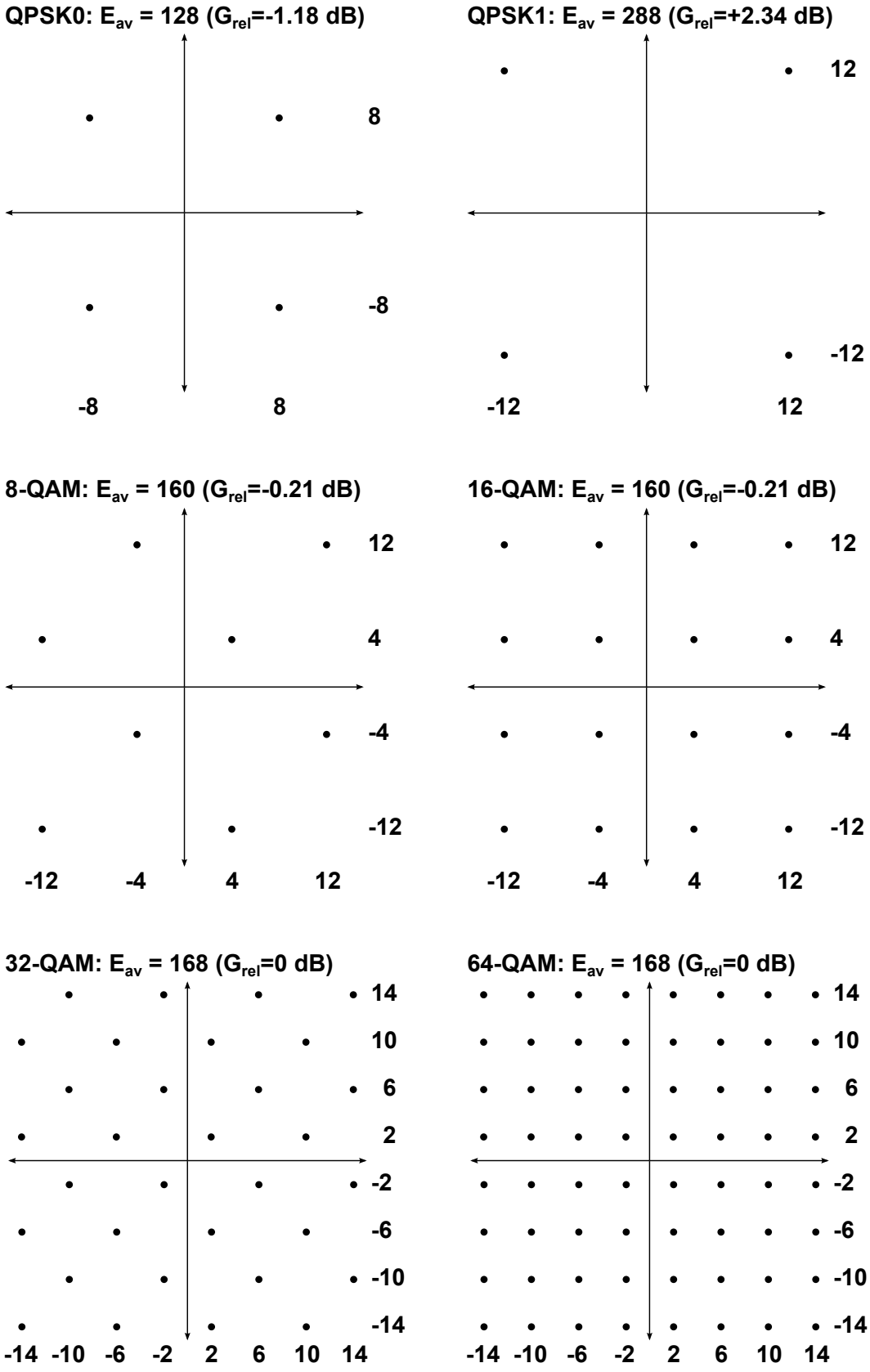


Figure 4.2: DOCSIS Upstream Modulation Modes

in the 5-bit version is simply a padded bit with a value of zero. This format is employed by DOCSIS because the upstream S-CDMA modes employing trellis-code modulation use modulation orders up to 128-QAM. The 5-bit format allows the 128-QAM constellation to span the same signal space as the lower order modulations.

4.2.2 Symbol Mapping

The data bits are Gray-code mapped according to Figure 4.3 [11]. The DOCSIS specification also allows differential encoding or trellis code modulation, however these encoding schemes are outside the scope of this work.

All of the DOCSIS maps shown in Figure 4.3 are available for use in the channel emulator. The QAM constellations with an odd number of data bits (i.e. 8-QAM and 32-QAM) have Gray-code violations, which are represented in the figure by dotted lines.

4.3 Determining Upstream RF Channel Emulator Characteristics

This section defines the capabilities of the channel emulator and are carefully chosen to meet the DOCSIS Physical Layer Specification [11]. Only those specifications relevant to this project are examined.

4.3.1 Carrier to Interference Plus Ingress Ratio

Cable networks must maintain their cable plant to comply with the DOCSIS standard. This means the quality of the signal has a carrier to interference plus ingress ratio of at least 25 dB. The interference plus ingress is explicitly defined as the sum of the noise, distortion products, and discrete and broadband ingress signals. In this project, noise and distortion are modelled with AWGN.

Ingress interference originates from nearby RF transmitters that are external to the network. This interference is unique to each cable network and is very difficult to emulate in a general way that is applicable to any network. Ingress noise is therefore not included in the emulator, and as such the carrier to interference plus ingress noise ratio is an SNR.

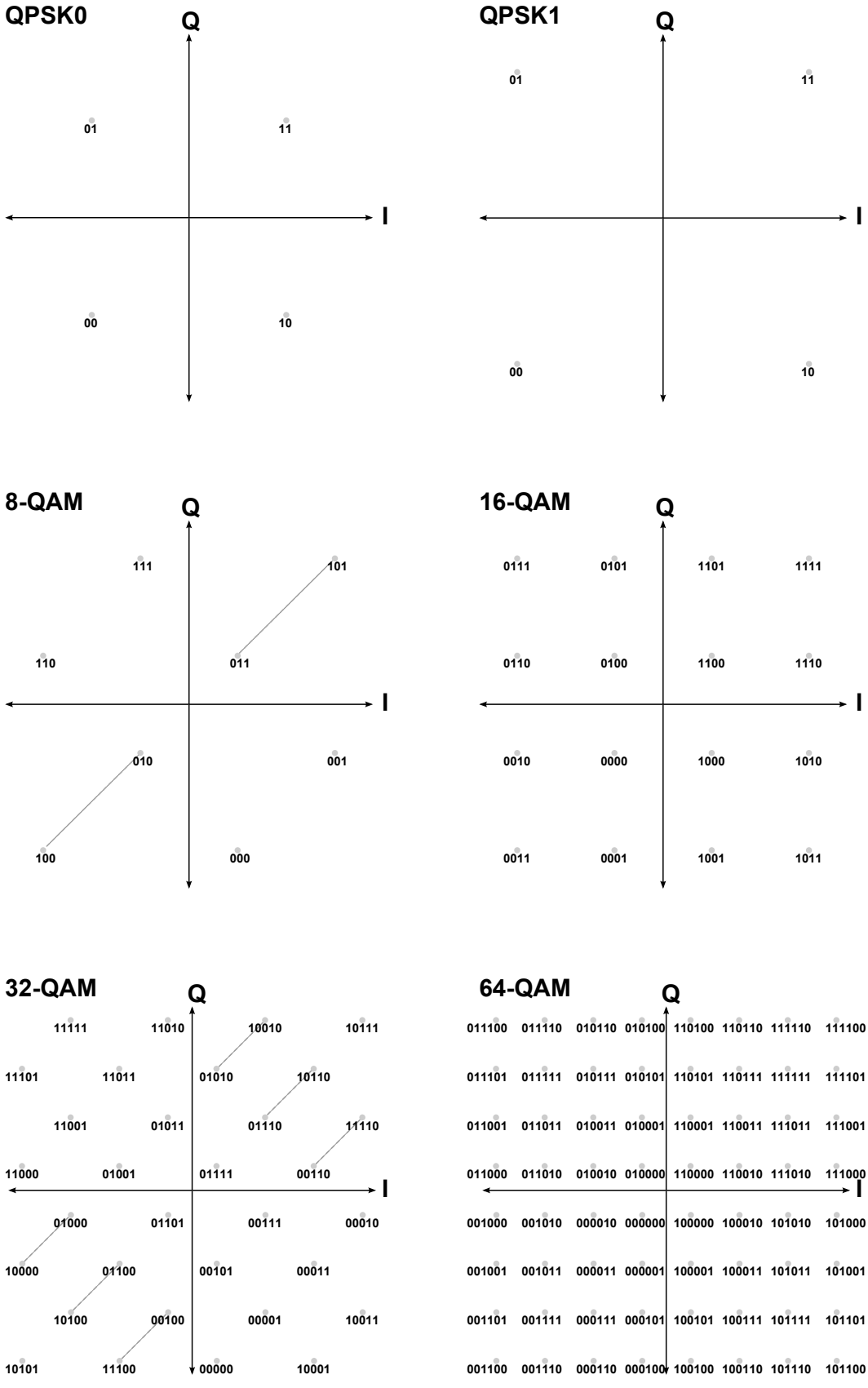


Figure 4.3: DOCSIS Upstream Symbol Maps

4.3.2 Determining Carrier Frequency Agility and Quality

The accuracy that is required for a numerically controlled oscillator (NCO) to be used as a carrier is specified indirectly. The upstream RF center frequency used by a particular CM is determined by commands received from the CMTS. If there is a discrepancy between the actual CM center frequency and the frequency assigned by the CMTS, a frequency offset command is issued by the CMTS to correct the error. These offset frequency commands have a resolution of 1 Hz. With this in mind, all NCOs used in the emulator must have a frequency resolution of 1 Hz.

The quality of the NCO to be used for a carrier is specified in terms of the phase noise it produces. Phase noise on the carriers is due mainly to timing jitter on the crystal reference oscillator. This timing jitter is directly related to the quality of the crystal oscillator and cannot be controlled from the emulator. However, phase noise is also produced by finite register lengths, and can be reduced to an arbitrarily low level by choosing an appropriate number of bits for the phase accumulator used in the NCO. To meet the carrier frequency agility and phase noise requirements a 32-bit phase accumulator is used for all NCOs in the emulator.

4.3.3 Micro-Reflections

The DOCSIS 3.0 standard states that a channel can contain up to 3 micro-reflections (echoes), with each being specified by its maximum strength and delay time. The strengths are given relative to the power in the carrier in dBc (defined in Equation 1.3), and the delay time is given in microseconds. The echo parameters for a DOCSIS upstream channel are provided in Table 4.1, where both the strength and delay time are referenced to the main path.

An RF signal on the upstream portion of the spectrum can contain zero, one, two, or three echoes. However, no two echoes can be from the same delay segment in Table 4.1. That is, multiple echoes with similar strengths and delay times are not allowed.

Phase shifts can be introduced by the impedance mismatches that cause the micro-

Table 4.1: DOCSIS Upstream Channel Micro-Reflections

Echo Number	Maximum Strength (dBc)	Delay Time, τ (μs)
1	-10	$0 < \tau \leq 0.5$
2	-20	$0.5 < \tau \leq 1.0$
3	-30	$1.0 < \tau \leq 1.5$

reflection. The impedance can be complex, having both resistive and reactive components. As a result, it is clear that phase shifts in the echoes can be anywhere from 0 to 1 cycle (0 to 2π radians). The channel emulator must include in effect a phase shift from 0 to 1 cycle along with micro-reflections.

4.3.4 Determining MER Perfomance of Emulator

It is the responsibility of the cable service provider to maintain the cable plant such that a certain quality of service is ensured. The signal processing algorithms used in DOCSIS compliant modulators and demodulators must function reliably in the presence of channel impairments. The DOCSIS standard specifies the quality of the signals delivered by the cable plant in terms of minimum MER.

Should a cable service provider wish to support a particular operation mode at a particular modulation rate, they must comply with the minimum MER values specified by DOCSIS for that operation mode and modulation rate. For the TDMA mode operating at a modulation rate of $F_s = 5.12$ Msymbols/second, which is the scenario of interest in this project, the minimum MER is specified as follows:

Case 1: Flat channel (i.e. no echoes) without Equalization

- Case 1a: MER $\geq$ 27 dB from 15 to 30 MHz
- Case 1b: MER $\geq$ 24 dB from 10 to 15 MHz and from 30 to 35 MHz
- Case 1c: MER $\geq$ 23 dB from 5 to 10 MHz and from 35 to 42 MHz

Case 2: Flat channel with Equalization

- Case 2a: MER $\geq$ 30 dB for QPSK
- Case 2b: MER $\geq$ 35 dB for all other modulations

Case 3: Echo channel with Equalization

- Case 3a: MER $\geq$ 30 dB for QPSK with only a single echo
- Case 3b: MER $\geq$ 33 dB for all other modulations with only a single echo
- Case 3c: MER $\geq$ 29 dB for all modulations with 2 or 3 echoes

The emulator conceived in this project must emulate channels that generate these MER values, which are a consequence of the impairments previously discussed. The implementations of the impairments must not introduce additional significant ISI or noise to degrade the MER from what is intended. That is, the impairment being modelled must be the dominant component of the distortion, not implementation-dependent limitations of the impairment.

For the purpose of algorithm development and implementation in this thesis, it is important to have a distortion budget for the emulator. Setting such a budget is as much an art as it is a science, but nevertheless cannot be overlooked. For the work in this project, the budget is set to allow at most 1% of the distortion to be inadvertently introduced due to the effects of finite register lengths. If the noise power generated by channel impairments is expressed as P_{chan} and the unintended noise power due to implementation is expressed as P_{imp}, the ratio P_{chan}/P_{imp} expressed in units of dB has

$$\text{Noise Power Ratio in dB} = 10\log\left(\frac{P_{chan}}{P_{imp}}\right). \tag{4.3}$$

A noise component that has 1% of the power of the desired signal component should not contribute significantly to the total power in the signal. In terms of SNR, this ratio corresponds to an SNR of 20 dB. The MER given by Equation 4.2 is itself a type of SNR, since it is the ratio of the average symbol energy to the average noise energy. This means

that in the absence of channel impairments, the MER must be 20 dB higher than the most stringent MER requirement (i.e. MER $\geq$ 35 dB for case 2b). Therefore, the MER due to implementation must be at least 35 dB + 20 dB = 55 dB.

4.4 Determining Pulse Shaping Filter Length

Ideally, the pulse shaping filter should be a square-root raised cosine (SRRC) filter with a rolloff factor of $\alpha = 0.25$. For a symbol rate of F_s, the magnitude response of a SRRC filter reaches zero at a frequency of $\frac{F_s}{2}(1+\alpha)$. This frequency establishes the bandwidth of the channel at

$$B = \frac{F_s}{2}(1+\alpha), \tag{4.4}$$

where F_s is the symbol frequency in symbols/second, α is the rolloff factor of the filter, and B is the bandwidth in Hz. For the specified symbol rate of $F_s = 5.12$ Msamples/second, this gives $B = 3.2$ MHz.

The impulse response of the ideal SRRC filter is infinite and so cannot be implemented. In reality, the impulse response must be truncated to a finite number of coefficients for implementation. This truncation reduces the filter performance in two ways: ISI is introduced by the modified filter and ripple is introduced in the magnitude response. Such filter defects reduce the MER that can be realized by a communication system, as examined in the next subsection.

4.4.1 Pulse Shaping Filter Length and MER

The length of the pulse shaping filter determines the maximum MER that can be achieved by the emulator. Since this filter length is directly related to the number of multiplies needed to compute each output sample, it is clear that a lengthy filter response requires more device resources and will be more costly to implement. An economical system will therefore contain a pulse shaping filter that is as short as possible while still meeting the best-case MER target determined in Section 4.3.4.

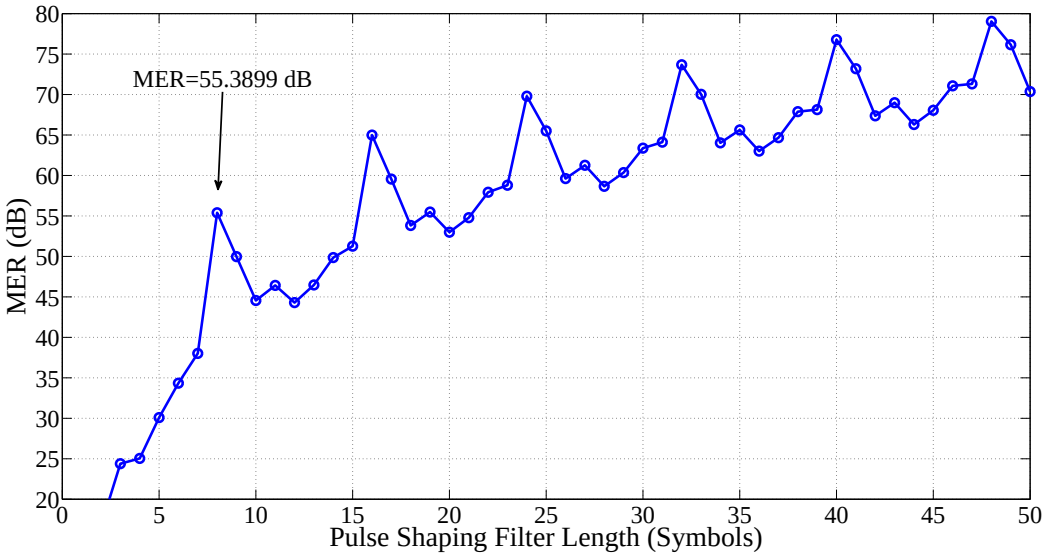


Figure 4.4: Effect of Pulse Shaping Filter Length on MER

In addition to the pulse shaping filter length, the length of the matched filter also plays a role in the maximum MER that can be achieved since the ideal channel response is the combined action of the pulse shaping and matched filter pair. In order to isolate the impact of the pulse shaping filter length on the MER, a matched filter with a length of 1000 symbols is used to simulate the ideal filter with infinite impulse response. The pulse shaping filter length versus MER curve is shown in Figure 4.4, which is computed with MATLAB for a sequence of 8000 QPSK symbols.

The shortest pulse shaping filter that achieves the MER target of 55 dB has a length of 8 symbols. Of course, there are longer pulse shaping filters that will exceed the 55 dB MER target, however the modest performance improvements of such filters do not justify their significant cost increases. On the basis of economy and MER performance, the 8 symbol pulse shaping filter is chosen for use in the main channel of the emulator.

4.4.2 Spectral Mask

A spectral mask is an envelope that provides the upper and lower bounds for the magnitude response of the pulse shaping filter. The DOCSIS 3.0 standard specifies 4 critical points in the spectral mask, which are given in Table 4.2. The symbol rate, F_s, is expressed in sym-

bols/second, and the frequency locations are given in units of Hz. This set of limitations, together with the MER, determine the minimum length of the pulse shaping filter.

Table 4.2: DOCSIS Upstream RF Channel Spectral Mask Limits

Frequency (Hz)	Lower Limit (dB)	Upper Limit (dB)
0 to $\frac{1}{4}F_s$	-0.3	+0.3
$\frac{1}{4}F_s$ to $\frac{3}{8}F_s$	-0.5	+0.3
$\frac{1}{2}F_s$	-3.5	-2.5
$\frac{5}{8}F_s$	—	-30

Spectral mask limits are imposed to ensure that interference from an adjacent channel does not significantly degrade the signal in the channel of interest (COI). In a channel emulator, the out-of-band specifications in the spectral mask do not apply to the COI as the adjacent channels are not demodulated. Only the center channel (i.e. the main channel) of the 3-channel FDMA spectrum produced by the emulator will be demodulated by the receiver under test. Accordingly, the two adjacent channels in the channel emulator must comply with the out-of-band emissions limits as they will affect the COI. Relaxation of the spectral mask limits in the COI allows the length of its pulse shaping filter, and therefore the cost of the emulator, to be reduced.

Another factor that affects the filter length is the desired level of the of the magnitude response at the channel edge. As outlined in the spectral mask, the magnitude response at the channel edge must be at least 30 dB below the level of the pass band. This spectral mask requirement is indicated by the dashed line in Figure 4.5, where a portion of the magnitude responses for 8 symbol and 24 symbol rectangular windowed pulse shaping filters are shown. Clearly the 8 symbol filter fails to achieve the required maximum channel width. Increasing the length to 24 symbols allows the pulse shaping filter to meet the spectral mask requirement, but comes with a threefold increase in cost.

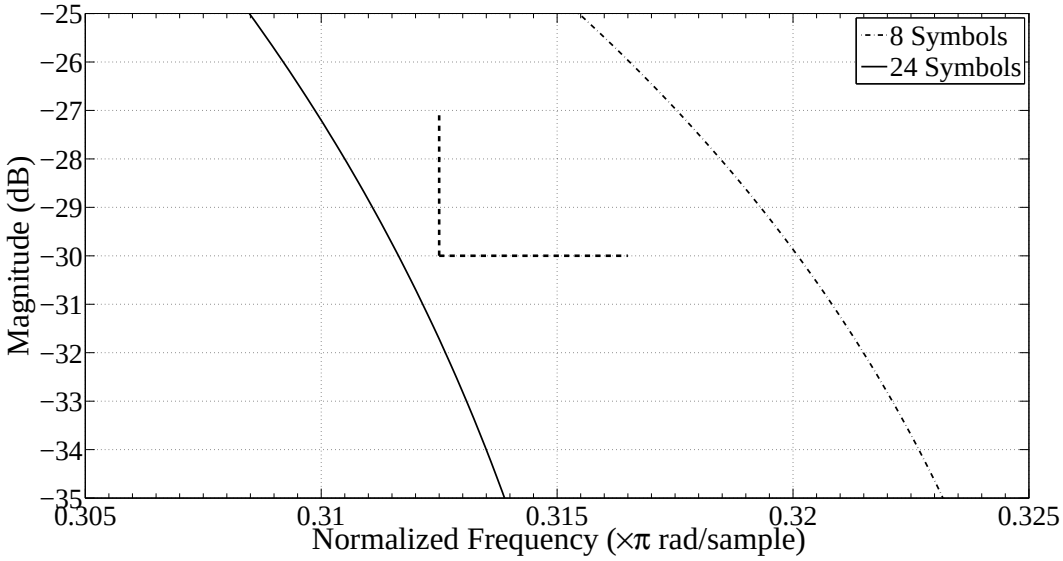


Figure 4.5: Effect of Pulse Shaping Filter Length on Bandwidth

4.4.3 Pulse Shaping of Adjacent Channels

The criteria for choosing an appropriate pulse shaping filter for the adjacent channels is somewhat different than for the channel of interest. Spectral leakage from the adjacent channels into the COI is the primary concern as it can adversely affect the MER. This means all specifications affecting out-of-band emissions must be met, including the -30 dB channel edge requirement of the spectral mask.

Since there are two adjacent channels implemented in the emulator, any increase in the filter length costs twice as much as it would in the COI. Unfortunately, the low cost 8 symbol filter used in the COI will not meet the spectral mask limits and a longer filter must be used. The shortest truncated (i.e. rectangular windowed) filter that meets the critical spectral mask limit is 24 symbols long, but only achieves a modest stop band attenuation of approximately 41 dB.

One way of increasing the stop band attenuation of a filter is to window the impulse response. Unfortunately, windowing widens the transition band of the filter [28] which has the appearance of increasing the rolloff factor, α. Knowing the transition band will be increased by windowing, the filter can be preconditioned by choosing a smaller rolloff

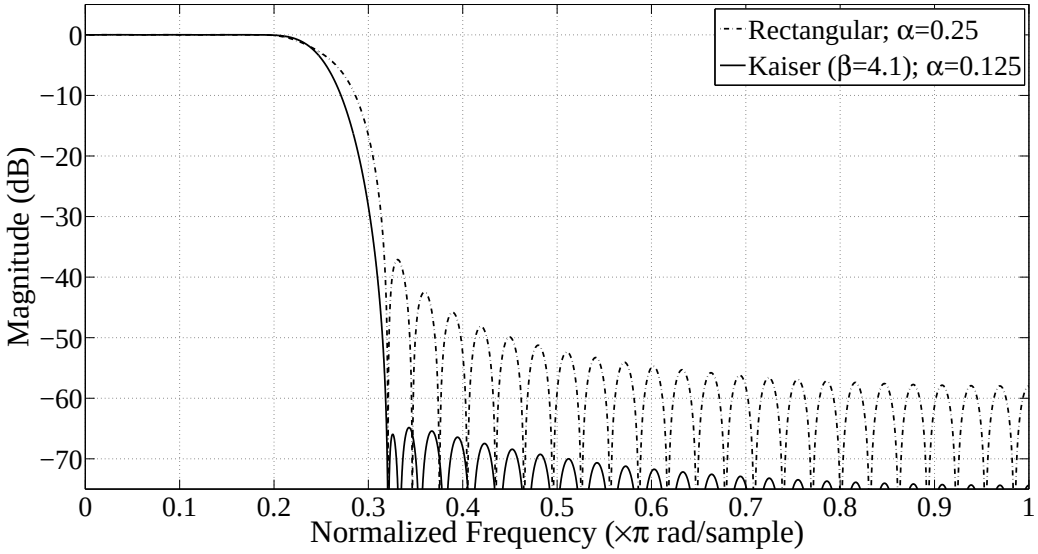


Figure 4.6: Modifying Adjacent Channel Pulse Shaping Filters

factor. With sufficient preconditioning, a strong windowing function can be applied to the impulse response to significantly increase the stop band attenuation. With a strong window a relatively short filter can meet the out-of-band emission specification. When done carefully, this has a minimal effect on both the width and the ripple of the pass band.

The magnitude response shown in Figure 4.6 is the result of trial and error. A rectangular windowed 16 symbol pulse shaping filter with a rolloff factor of $\alpha = 0.25$ has a stop band attenuation of approximately 37 dB, as shown by the dotted line. After changing the rolloff factor from $\alpha = 0.25$ to $\alpha = 0.125$, a Kaiser window with a shape factor of $\beta = 4.1$ is applied to obtain the magnitude response shown by the solid line. The resulting filter meets all spectral mask limits and achieves a stop band attenuation of approximately 65 dB.

5. Practical Hardware Limitations

A purely mathematical simulation of the channel emulator yields results that do not account for all of the performance limitations in a hardware implementation. Compromises are often made in order to simplify the problem and reduce the hardware costs of the circuit. Tests must be performed to assess the impact of these compromises on the performance of the implemented circuit. To that end, this chapter consists of two parts. The first part explains the compromises made in the implementation, the reasons for them, and how these compromises are expected to affect the output. The second part of the chapter explains the test assembly used to verify the channel emulator. The results of performance verifications made on a couple of key circuits are then presented along with the end-to-end channel emulator performance.

5.1 Implementation Issues

The implementation issues presented in this section are due to limitations of the various equipment components used for implementation and verification. Some of these issues are inherent to the development board chosen for implementation. The channel emulator presented in this thesis is implemented on an Altera DE4 development board. Altera produces a few different configurations of this board using different on-board FPGA devices. The DE4 board used in the implementation contains a Stratix IV FPGA with model number EP4SGX530KH40C2. For more information on these Altera products, please refer to the Altera website[1].

[1] www.altera.com

Symbol Rate

DOCSIS 3.0 specifies the highest symbol rate to be 5.12 Msymbols/second, which is based on the use of a 10.24 MHz reference clock. The DE4 board on which the project is implemented does not have crystal reference clocks that are any multiple of the 10.24 MHz reference frequency — either a 50 MHz or a 100 MHz clock can be used as a reference. The PLL circuits inside the FPGA do not have the required resolution to be able to accurately produce a 10.24 MHz clock from these reference clocks. Because the design consists of several interdependent clock domains, the implemented symbol rate is 5 Msymbols/second. This rate, and the required multiples thereof, can be achieved exactly by the PLL on the FPGA.

Modification of the symbol rate has no effect on the function of the hardware but does affect the spectrum of the output. As 3.1 is dependent on the symbol rate, the bandwidth of the channels is altered from that specified by DOCSIS. The specified channel width of 3.2 MHz is therefore changed to 3.125 MHz when a symbol rate of $F_s = 5$ Msymbols/second is used. Additionally, the spectral mask frequencies given in Table 4.2 are also altered. All functionality can be verified with this modified symbol rate.

Digital to Analog Conversion

The complex channel output is passed to the 14-bit digital to analog converters (DAC) for verification on test equipment such as spectrum analyzers and vector signal analyzers (VSA). As the DAC only has 14-bit resolution, MER measurements obtained with the VSA are lower than those obtained by full-resolution analysis (i.e. no word length truncation) in MATLAB. Furthermore, the hold action of the DAC distorts the channel emulator output by acting as a low pass filter. Pre-distortion filters are often employed to counteract the rolloff [29], which can be done digitally before passing the signal to the DAC [30]. Due to the increased hardware cost and the fact that MATLAB can provide a more precise analysis, no pre-distortion filter is implemented.

Vector Signal Analyzer Limitations

The HP 89410A VSA is a useful piece of equipment for obtaining information about the channel in near real-time, but has some shortcomings which limit its ability to measure all aspects of the channel emulator performance. The RF front-end of this VSA can only accommodate a spectrum from 0 to 10 MHz. Inclusion of adjacent channels in the signal significantly reduces the quality of the demodulation as the total bandwidth exceeds the 10 MHz limit. Furthermore, any carrier mismatch created by the local oscillator in the VSA adds more noise to the demodulated signal that cannot be directly controlled.

There is an adaptive equalizer circuit inside the VSA which is useful for correcting the echoes generated by the channel emulator. This equalizer is similar to the one implemented in the receiver under test with one important distinction — it does not generate complex equalization coefficients. The consequence is that the equalizer in the VSA can only correct the real component of the echoes. Echoes with a purely complex phase shift are unaffected by the equalizer, making the VSA inadequate for verifying such cases.

Despite these limitations, the VSA is still a valuable piece of test equipment since any changes made to the channel conditions can be quickly verified. It may not be the best choice for quantifying the performance of the channel emulator, but its ability to provide snapshots of the channel output in near real-time makes it an excellent tool for qualitative analysis of the circuit output.

5.2 Test Assembly and Hardware Performance Results

A dual approach is taken to the testing of the channel emulator. Quantitative performance is calculated offline by capturing the output on RAM blocks instantiated in the FPGA, converting the files to a MATLAB-friendly format, and importing to MATLAB for analysis. Although this method achieves the best precision it is very time consuming. It is only used when precise MER measurements must be made. However, there are often times when precise MER measurements are not needed. In such cases the qualitative results provided by the VSA are adequate for diagnosing the channel emulator output. The qualitative and

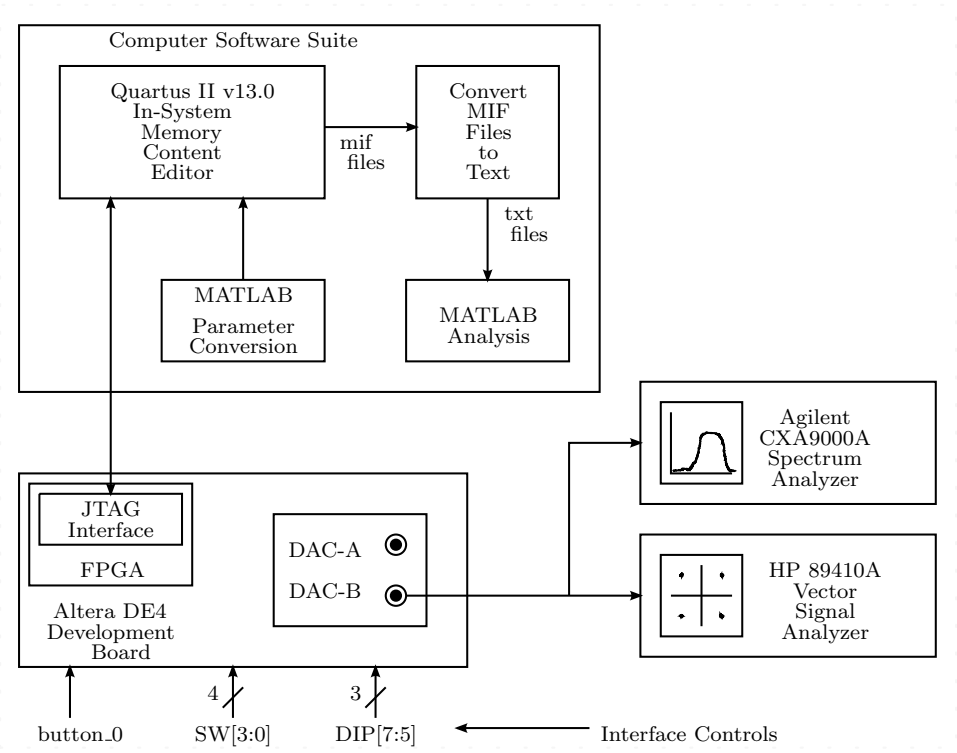


Figure 5.1: Schematic for Channel Emulator Test Assembly

quantitative verification approaches are both satisfied by the channel emulator test assembly shown in Figure 5.1.

Verifying Fractional Delay

The functionality of the fractional delay filter is easily verified in simulation since it is an FIR filter with only 3 coefficients. However, it is just as important to verify the output on test equipment to ensure the desired effect is achieved. In the case of the fractional delay, it is necessary to verify that the fractional delay input parameter is being correctly interpreted by the circuit.

In order to do so, a pulse shaped and upsampled signal is generated and passed to two fractional delay filters in parallel. The fractional delay input to the first filter is set to zero (i.e. no delay), and the input to the second filter is set to some non-zero value. Each output is passed to a DAC and sent to separate channels on an oscilloscope for time domain verification.

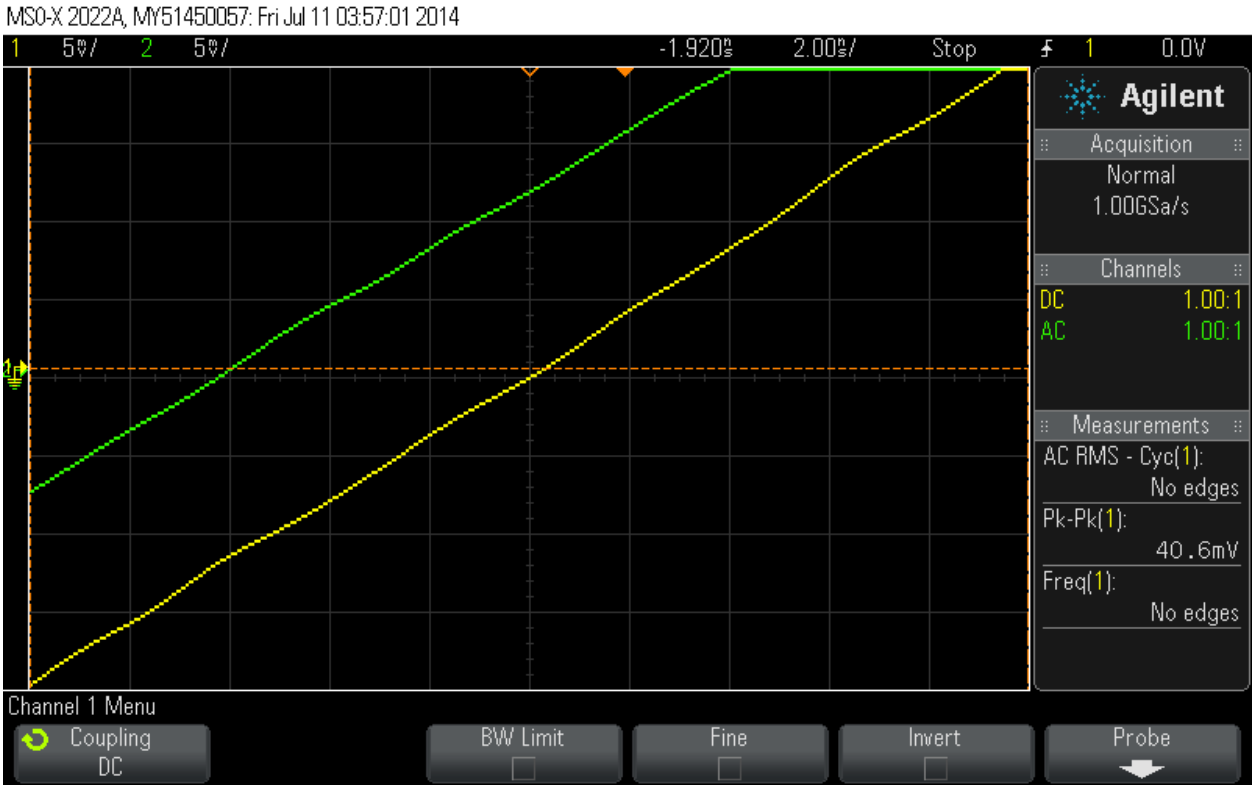


Figure 5.2: Functional Verification of Fractional Delay Filter

An example of such a test is shown in Figure 5.2. The bottom trace passing through the center of the plot is the zero delay reference trace, and the top trace is the fractionally delayed signal. The sample rate for these signals is 40 Msamples/second, which has a sampling period of 25 nanoseconds. The delay setting for the top trace is -0.25 samples corresponding to a time advance of 6.25 nanoseconds, which is confirmed by Figure 5.2.

Verifying AWGN Generator Performance

Although the spectrum analyzer is useful for making noise power measurements, it is inadequate for verifying the probability distribution of the noise samples. Both the Gaussian noise and the uniform random variables from which the Gaussian noise is derived have a flat spectrum, making them appear identical on the spectrum analyzer. In this case, noise samples must be captured in RAM and converted for analysis in MATLAB.

For each of the complex noise components, 65,000 samples are captured and analyzed.

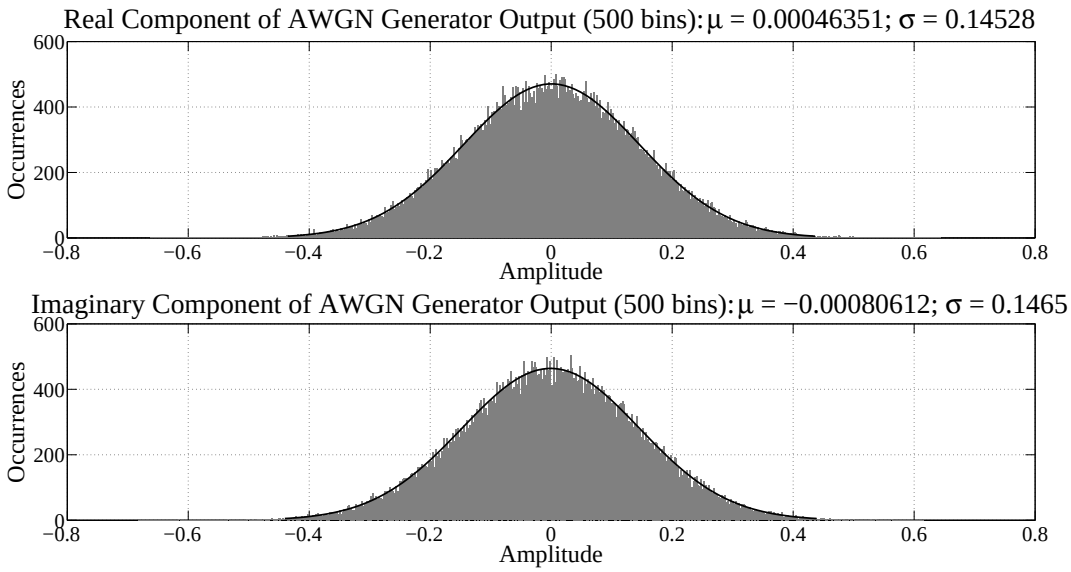


Figure 5.3: Histogram of Hardware AWGN Generator Output

A histogram with 500 bins is generated for each component, and a Gaussian curve is fitted to the histogram. The mean, μ, and standard deviation, σ, are calculated for each and displayed above the plots shown in Figure 5.3. These plots verify that the complex AWGN samples being generated have a distribution that is very nearly Gaussian, with the mean of each component being very close to zero.

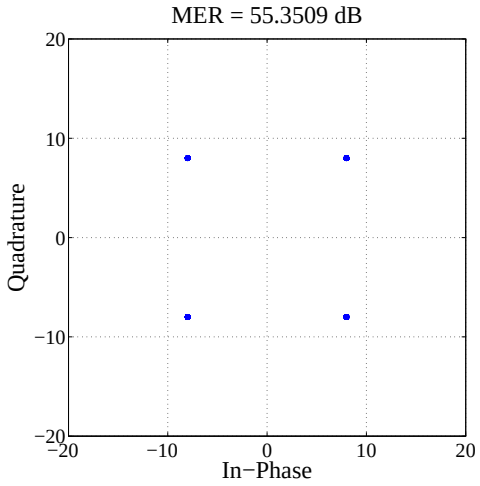
The second step of the noise verification involves making measurements to ensure that the SNR is correct. The output level of the AWGN generator is determined from the SNR and must be measured to ensure its accuracy. The Agilent CXA 9000A spectrum analyzer is used to measure the signal power of the main channel across the entire spectrum without noise present. Frequencies below 1 MHz cannot be measured as the baluns that lie between the DAC and the spectrum anlayzer do not pass these frequencies. To circumvent this limitation the main channel is upconverted to a center frequency of 5 MHz, and the measurement span is from 1 to 41 MHz (i.e. from 1 MHz to $F_{samp} + 1$ MHz, where $F_{samp} = 40$ MHz). Once the signal power across this band has been measured, the channel gain is set to zero, the noise is toggled on, and another measurement is made. The difference between the two levels gives the SNR. These measurements are made at a high channel gain of +18 dB to ensure the noise level of the AWGN is far above the noise floor of the spectrum analyzer.

Measurements taken for a signal with a center frequency of 5 MHz verify the SNR for each modulation scheme is within ±0.1 dB of the setting.

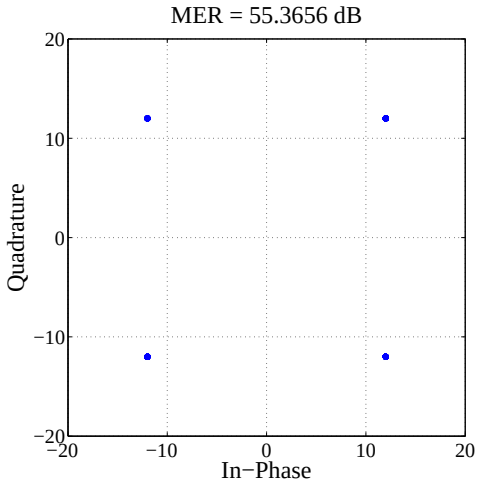
Channel Emulator Performance

The best-case channel emulator performance is measured using captured data and MATLAB. Data is first collected in RAM for a baseband channel with no echoes, no noise, and no adjacent channel interference. The captured complex signal is match filtered and decoded inside MATLAB. Likewise, the data symbols generated by the circuit are captured in another RAM and passed to MATLAB. These symbols are mapped to constellation points inside MATLAB to create reference points for the MER calculations.

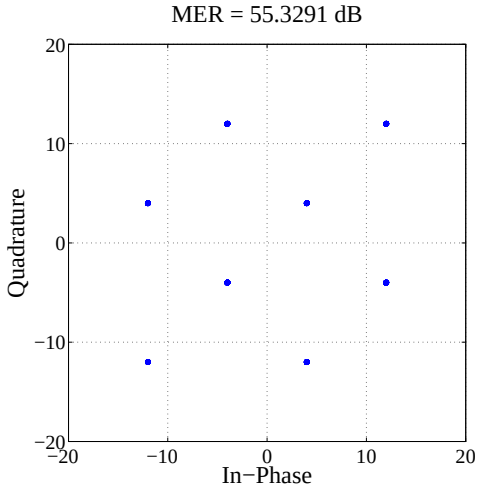
With the preamble symbols removed, the pair of RAMs that capture the complex data can accommodate approximately 8000 data symbols for one MER analysis. Best-case performance is measured for each of the 6 modulation modes, the results of which are shown in Figure 5.4. The MER for each case is displayed above the corresponding constellation plot. As can be seen, the MER performance goal of 55 dB that was set in Chapter 4 is achieved for all modulation modes.



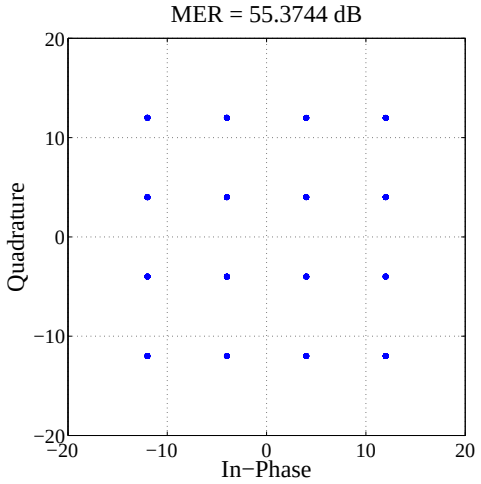
(a) QPSK0



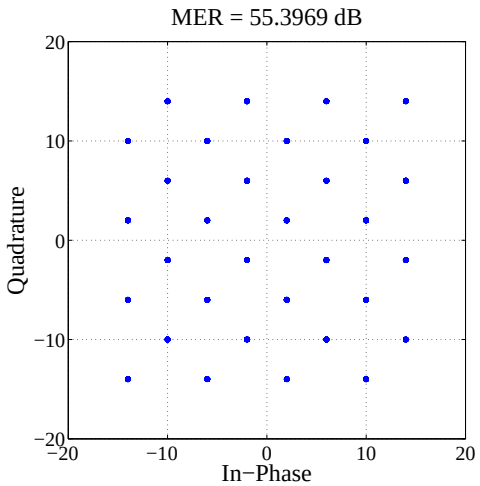
(b) QPSK1



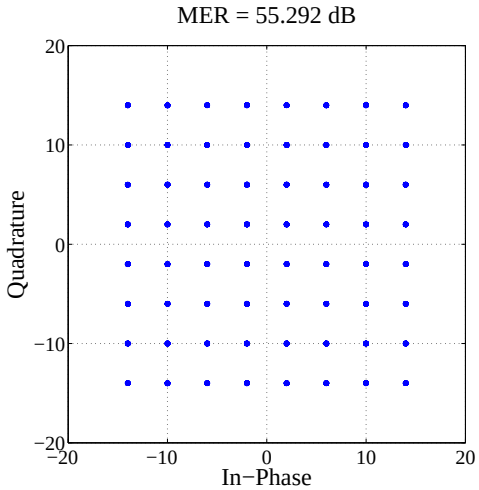
(c) 8-QAM



(d) 16-QAM



(e) 32-QAM



(f) 64-QAM

Figure 5.4: Unimpaired Channel Output Results

6. Conclusion and Future Work

6.1 Summary

The design for a realistic yet efficient hardware channel emulator is complex. The design involves the integration of many different functions to implement a modulator and channel emulator into a single unit. One must ensure that the performance specifications are met without over-achieving them as doing so costs hardware. Since multipliers are the most costly elements in an FPGA device, multiplier-efficient algorithms must be derived and carefully implemented in every function to optimize cost. Implementation of cost-efficient hardware is meticulous and arduous, especially in the debugging phase.

In this thesis, the methodology for implementing an economical DOCSIS 3.0 channel emulator is outlined. The intent is to provide a procedural explanation of the design process while justifying the hardware structures used for implementation. Certain key specifications from DOCSIS 3.0 are presented as they pertain to the capabilities of the emulator. The functional verification is performed and followed by a channel emulator performance measurement of the best-case MER achieved without any channel impairments. This chapter concludes the thesis by summarizing the important results and contributions, and outlines future work that could be done to improve the efficiency and functionality of the channel emulator.

6.2 Contributions and Results

The main contribution of this work is the design and implementation of multiplier-efficient circuits. This thesis focusses on integrating the many circuits that implement a channel emulator combined with a QAM modulator. The combining of the emulator and modulator

enables a baseband emulation of channel impairments normally encountered after upconversion to RF. The advantage to this approach is that the implementation is done at a much lower sampling rate, which allows multipliers to be time-shared to a higher degree and thus greatly reduces the cost.

Channel emulation with a digital circuit located in the FPGA that implements the receiver combines economy with reduced receiver development time. Receiver testing is not dependent on the analog front-end being functional which allows the receiver algorithms to be tested earlier in the development process. This enables the DSP portion of the receiver to be developed concurrently with the analog front-end, thereby reducing the time-to-market. Furthermore, the emulator/modulator structure can be instantiated as necessary by many designers working independently on different receiver configurations, thus eliminating the expense of a large number of commercial channel emulators.

6.3 Resources and Performance

The DOCSIS-compliant emulator architecture presented in this thesis provides a balance between flexibility, performance, and economy. Any set of channel conditions in a DOCSIS 3.0 governed CATV system can be emulated. Three streams of data are modulated independently to produce a 3-channel signal, with the main channel being flanked by two adjacent channels. The main channel contains up to 3 micro-reflections. Complex AWGN is generated and injected to emulate thermal noise in the amplifiers of the receiver.

In terms of device resources, the combined emulator/modulator structure uses 33,658 combinational lookup tables (LUT) of the 424,960 available on the Stratix IV EP4SGX530KH40C2 device. This constitutes approximately 8% of the available LUTs. There are 30,295 dedicated logic registers used, constituting approximately 7% of the available registers. The design also uses 43 dedicated multipliers of the 1,024 available multipliers — approximately 4%. The FPGA block RAM used by the combined emulator/modulator structure is 401.125 KB of the available 2592 KB — a 15% usage. Finally, only a single phase-locked loop (PLL) was required to generate the necessary clock signals. As there are 8 PLLs available on the device, this constitutes approximately 13% usage.

The MER is used as a performance measure for the emulator. As stated in Chapter 5, the emulator achieves an MER of 55.29 dB for all 6 modulation modes, thus satisfying the performance goal of 55 dB which was set in Chapter 4.

6.4 Future Work

Implementation of the Box-Muller algorithm for AWGN generation involves a trade-off between economy and the quality of the Gaussian approximation. Further hardware reduction could possibly be realized in the Box-Muller AWGN generator by replacing the successive approximation circuits with ROMs containing precomputed values of $f(x_1)$ and $g(x_2)$. In this case additional control logic is needed for ROM addressing. This method is described in [31]. The implications of using a lower quality Gaussian approximation for thermal noise emulation were not investigated.

The design of the transmit data circuit presented in this work only allows the same modulation type used in the data packet and the preamble. For example, no provisions have been made to allow a QPSK preamble to be combined with a 16-QAM data packet. According to the DOCSIS 3.0 specification, the preamble must use either low-power or high-power QPSK modulation modes [11]. Previous DOCSIS 1.x specifications must use a 16-QAM preamble when a 16-QAM data packet is transmitted. Future modifications could be made to include independent modulation schemes for preamble and data packets.

A. Auxiliary Circuit Schematics

Circuit schematics presented in this chapter are graphical representations of the actual hardware implementations contained in the channel emulator. Only key components were presented in the body of the thesis in order to maintain continuity. However, many other important circuits are required for correct operation of the emulator. The remaining schematics are presented in this chapter along with brief explanations of the hardware operation. Reasons for choosing particular structures are also provided.

A.1 Generating and Mapping Symbol Data

Once the emulator has loaded the channel parameters from memory, the first step in the signal chain is symbol data generation. At the heart of the data generator used in the emulator is a set of linear feedback shift registers (LFSR) which generate the independent bits that make up a data word. These data words, generated at the symbol rate, are mapped to I-Q constellation points and form the basis for the signal used in the pulse shaping filter. Before examining the symbol generator, a brief summary of LFSRs is presented.

A.1.1 Linear Feedback Shift Register

As LFSRs are widely covered in literature only a brief review of their function is offered here. Linear feedback shift registers employ shift registers with taps at specific locations which feedback through some combinational logic (exclusive-or (XOR) gates in this case) back to the input. At every clock edge the oldest shift register contents are shifted out and a new feedback value is shifted in. The LFSR is loaded with a seed value upon power-up/reset and generates the same sequence each time as long as the seed value remains unchanged.

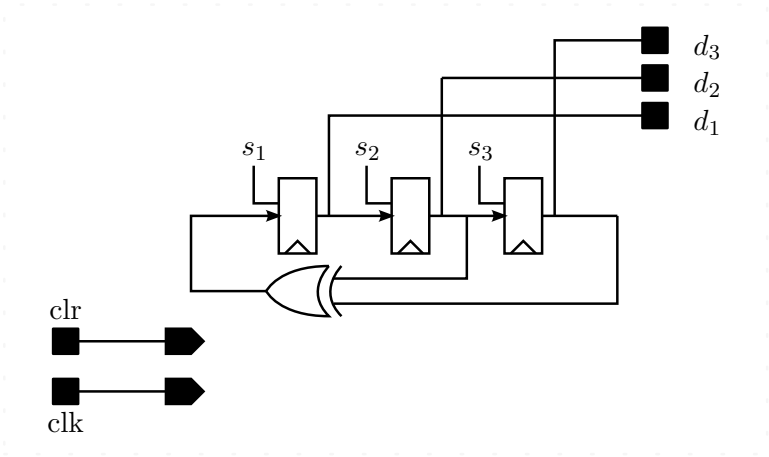


Figure A.1: Schematic for 3-bit Linear Feedback Shift Register (LFSR)

The sequence of values produced by an LFSR is pseudorandom — it is a finite length sequence with deterministic values that eventually repeat. If a maximum length sequence is used, the number of unique values generated by an LFSR is $2^M - 1$, where M is the number of bits in the LFSR. The LFSR will cycle through all possible values before repeating the sequence. The only exception is the value zero (i.e. all registers are zero simultaneously). This is referred to as the lock-up state, since the feedback network will always provide zero as an input if all registers are zero. Due to this lock-up state, an LFSR produces one less zero than ones in the bit sequence. For long shift registers, the bits can be considered to be of equal probability.

Consider the LFSR of Figure A.1. If seeded with the value {`s_3,s_2,s_1`}=`3'b101`, 3-bit words will be generated in the following sequence: 101,110,111,011,001,100,010. There are $M = 3$ bits in the LFSR, giving a sequence length of 7. Comparatively, LFSRs as long as 41 bits are used in the symbol generator, and lengths as high as 79 bits are used in the AWGN generator. They are used in parallel to generate multi-bit data words, with the output of each LFSR contributing a single bit to the data word.

A.1.2 Symbol Generator

The data words for the different modulation schemes can be from 2-6 bits in length. In order to increase independence between bits, each of the 6 bits generated for a word is

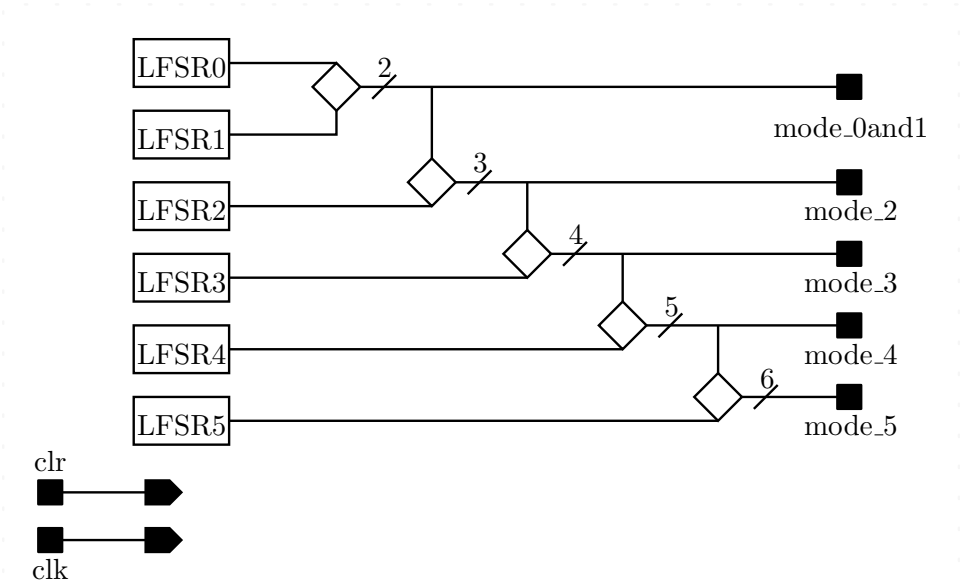


Figure A.2: Schematic for Symbol Generator

generated from an LFSR with a different length, and therefore a different period of repetition. The 6 LFSRs of various lengths each generate a single bit with every clock cycle, which are concatenated into 2, 3, 4, 5, or 6-bit words according to modulation scheme. The modulation selection parameter is provided by the load profiles circuit.

The LFSR lengths used for the symbol data words are 32, 33, 35, 36, 39, and 41 bits. These LFSRs are based on maximum length sequences provided in [32]. Unique seeds are used for each LFSR in each of the 3 symbol generators. Furthermore, unique wiring (i.e. which LFSR output is connected to which bit in the data word) is used for each of the symbol generators to further randomize data between channels.

A.1.3 Symbol Mapper

Each of the 2, 3, 4, 5, and 6-bit data words is mapped simultaneously according to the 6 modulation modes provided by the emulator. The I and Q components from each modulation map are passed to output data selectors whose modulation mode parameter is provided by the load profiles circuit. Immediately upon changing the modulation mode, the I-Q outputs are updated to the new modulation mode.

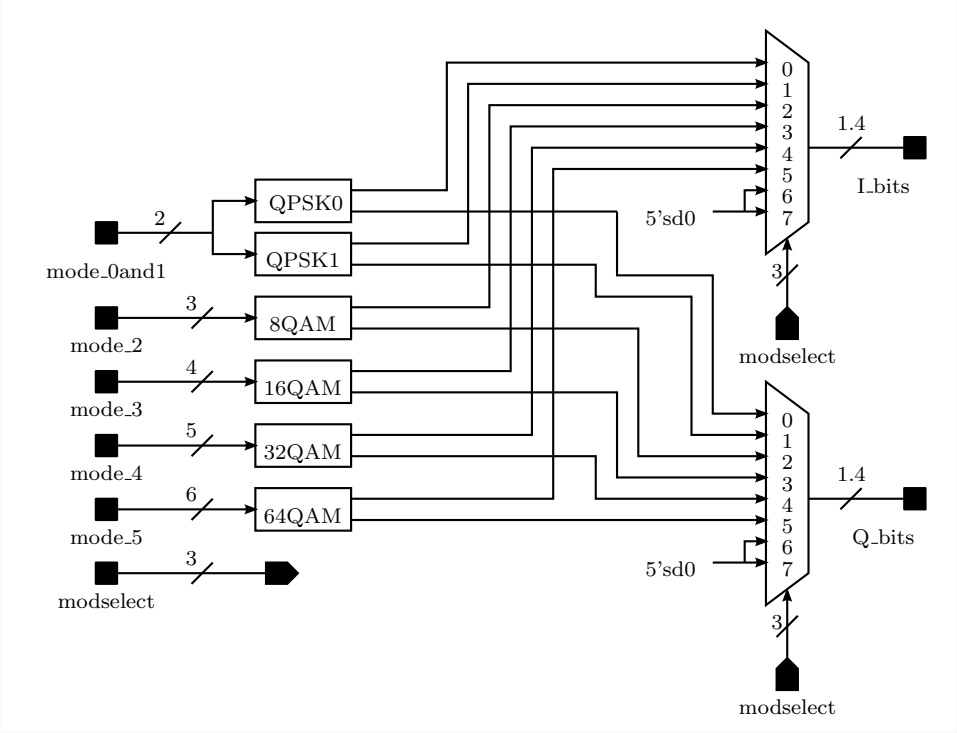


Figure A.3: Schematic for Symbol Mapper

The modulation maps used in the symbol mapper consist of combinational logic which encodes the input data to discrete locations on the complex I-Q plane. Specifically, each of the symbol maps concatenates alternating bits of the input to be used as select signals for a pair of output data selectors. The data inputs to these data selectors are lookup tables (LUT) which store the quantized I and Q amplitude values for the particular modulation mode. An example of such a symbol map is provided in Figure A.4, where the 16-QAM symbol map is shown. For the sake of brevity, only the 16-QAM map schematic is explicitly shown as all other symbol maps follow the same approach.

A.2 Multiplexed Sample Interleaving

As mentioned previously, multiplexed hardware structures reduce hardware requirements by time sharing multipliers and adders. When there is no sample rate change in the signal processing chain, a multiplexed structure with interleaved I-Q samples as an input will produce interleaved output samples in the same order. However, structures which perform

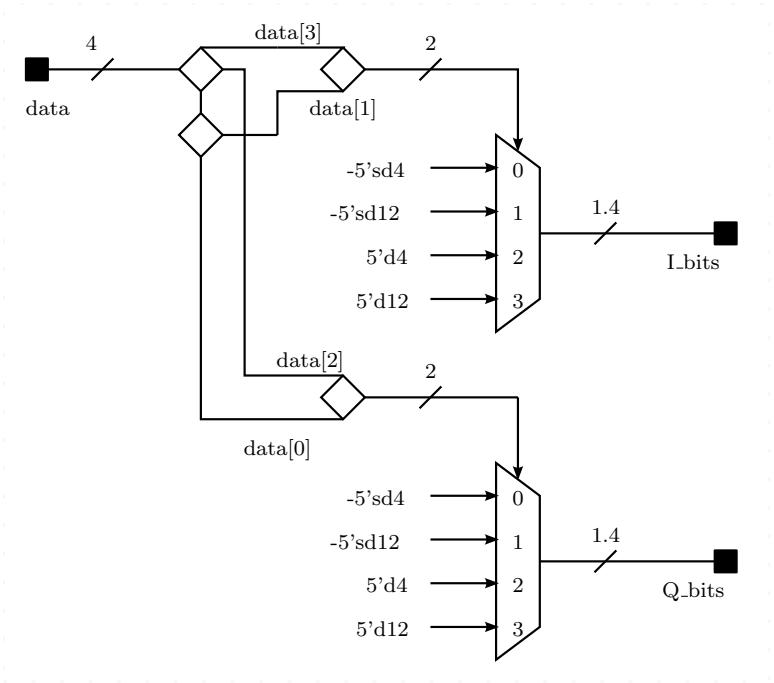


Figure A.4: Schematic for 16-QAM Modulation Map

upsampling on the multiplexed signal complicate the situation, as the proper sample order is no longer maintained at the output. Another component is required to reorder the output samples correctly. This section discusses sample interleaving in the channel emulator.

A.2.1 Interleaving the Pulse Shaping Filter Output

At the pulse shaping filter input, parallel I-Q streams are multiplexed to form a single input data stream. The multiplexed filter structure maintains the I-Q interleaving internally, but at the output the sample ordering is incorrect. The reason for this is the 4× upsampling action of the pulse shaping filter. For each input sample, 4 output samples are created. If a multiplexed I-Q pair of input samples is filtered, 8 output samples are produced — 4 I samples followed by 4 Q samples. As it is much more difficult to develop multiplexed hardware structures that process the 4 I, 4 Q multiplexing scheme than it is to develop structures with simple I-Q interleaving, an output interleaver is used after the pulse shaping filter to restore the correct sample order. The schematic for the pulse shaping filter interleaver is shown in Figure A.5.

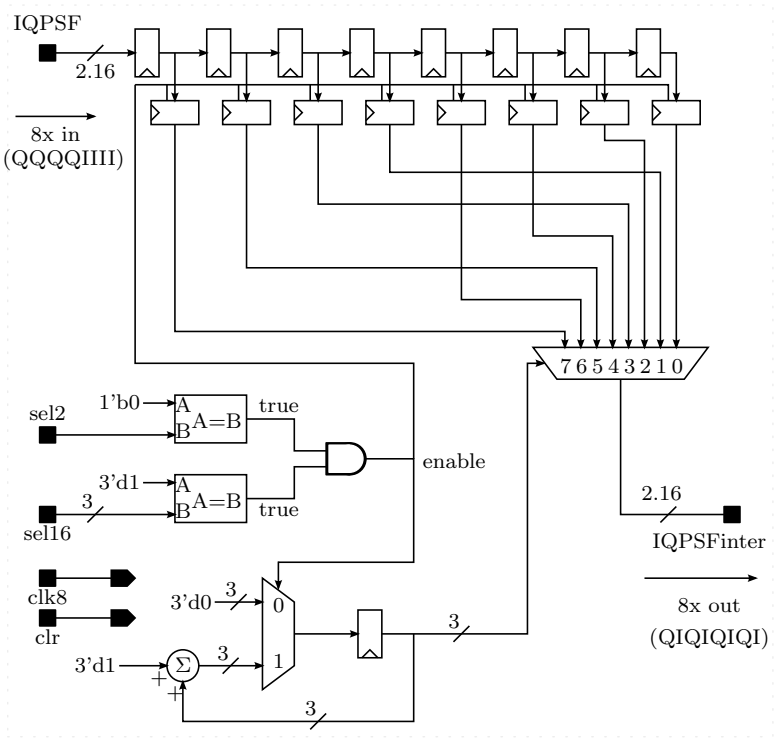


Figure A.5: Schematic for Pulse Shaping Filter Output Interleaver

A.2.2 Interleaving the Upsampling Filter Output

As with the pulse shaping filter, the combination of upsampling and multiplexing requires the output of the upsampling filter to be interleaved before further processing. The input to the upsampling filter is the interleaved pulse shaping filter output, arriving at a rate of 8 samples/symbol. The output from the upsampling filter operates at a rate of 16 samples/symbol. However, this output consists of 2 in-phase (I) samples followed by 2 quadrature (Q) samples which must be interleaved before continuing through the signal processing chain. This interleaver is depicted in Figure A.6.

The interleaver is constructed from a shift register operating at 16 samples/symbol which holds the throughput data. When `sel16[1:0]=2'd2`, the output registers are enabled. These registers take a snapshot of the shift register contents and hold them until the next snapshot. While the shift register collects the next 4 samples, an output multiplexer is used to interleave the samples held in the output registers to yield the correct order. The same signal used to synchronize the enables of the output registers is also used to cycle through the output

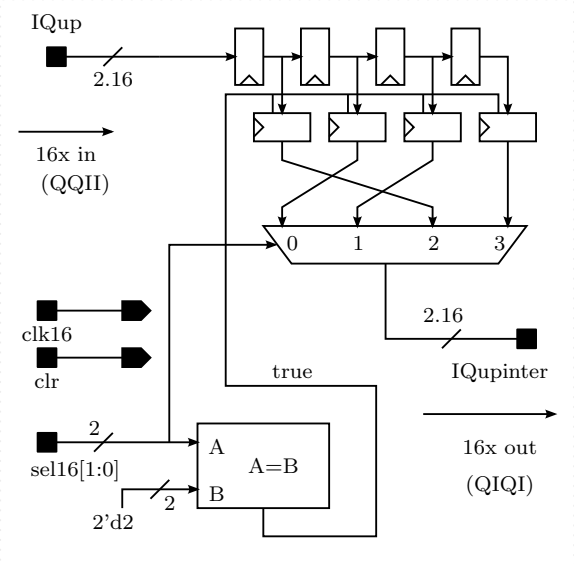


Figure A.6: Schematic for Upsampling Filter Output Interleaver

multiplexer data streams.

A.3 Upsampling Filter Assembly

The upsampling filter assembly is relatively straightforward. The multiplexed upsampling filter takes multiplexed I-Q samples at 8 samples/symbol as input, and produces output samples at a rate of 16 samples/symbol. The filter output sequence is two I samples followed by two Q samples, etc. This output is then interleaved by the upsampling filter interleaver which restores the correct multiplexed sequence of one I sample followed by one Q sample, etc. The upsampling filter assembly is shown in Figure A.7.

A.4 Echo Path Assembly

The echo paths of the emulator have only two functions: to delay and attenuate the input. However, since the path attenuation can be specified with a complex attenuation parameter, phase shift can also be induced on each echo path. The structure of the echo paths is shown in Figure A.8.

To use multipliers and adders more efficiently, the echo paths are implemented with multiplexed structures. Each echo path has 3 distinct input parameters: integer delay, fractional

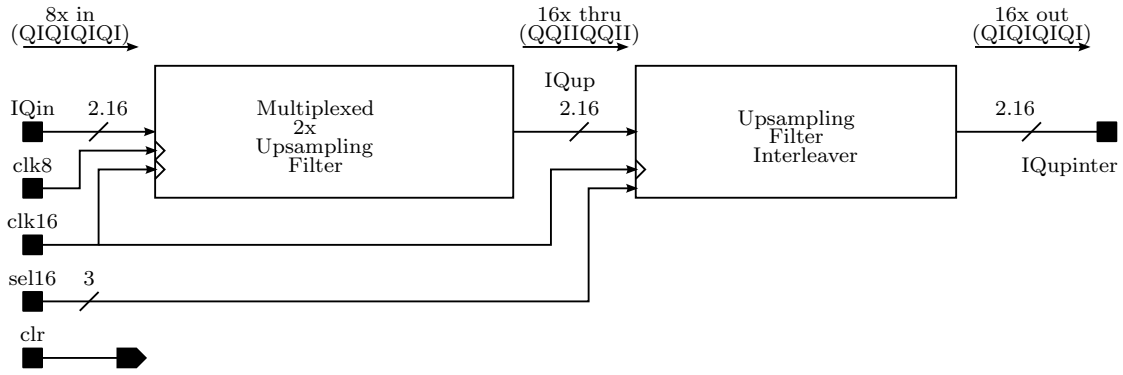


Figure A.7: Schematic for Upsampling Assembly

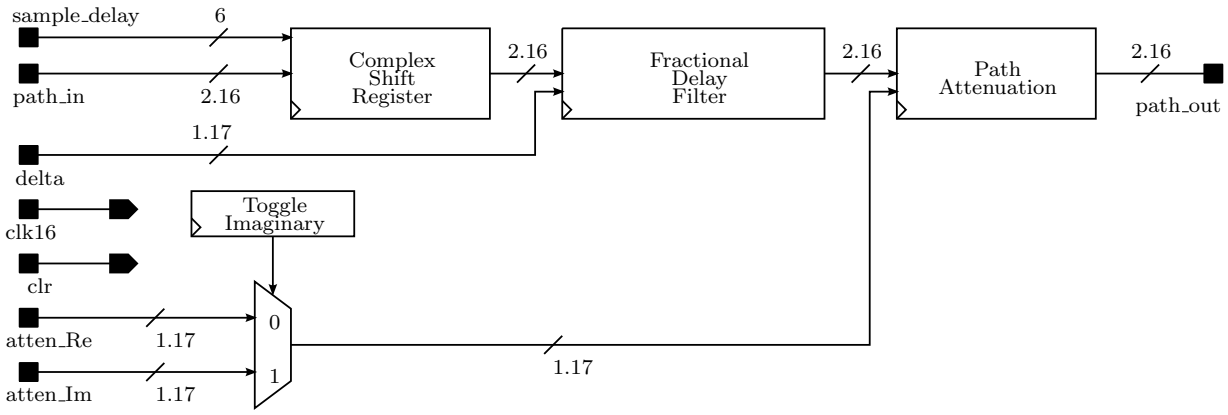


Figure A.8: Schematic for Echo Path with Complex Attenuation

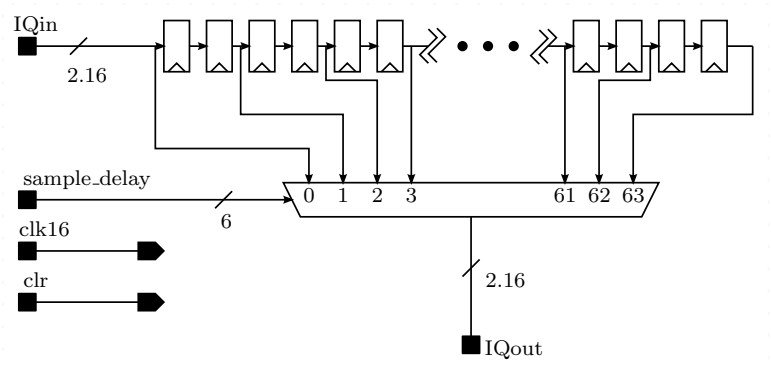


Figure A.9: Schematic for Multiplexed Complex Shift Register

delay, and path attenuation. As the fractional delay was already discussed Chapter 3, only integer delay and attenuation are addressed here.

A.4.1 Integer Delay

An integer delay is one of the simplest digital circuits to implement as only a shift register is required. Digital circuits are naturally suited to introducing a delay that is an integer multiple of the sampling period. For example, if $\mathbf{p}[n]$ is clocked through n_o registers in series, the delayed signal is $\mathbf{p}[n - n_o]$, where n_o is the delay in samples and of course must be a non-negative integer.

The circuit is multiplexed and upsampled to accommodate the complex baseband signal at 8 samples/symbol. The output multiplexer selects the appropriate number of samples of delay, which can be from 0 to 63 samples. For operation at 8 samples/symbol (25 ns sampling period), the maximum integer delay is 63 samples $\times$25 ns/sample = 1575 ns. This maximum integer delay covers the possible integer echo delay times as specified by DOCSIS 3.0. The schematic for the complex shift register is shown in Figure A.9.

Shift registers are first in first out (FIFO) circuits and as such do not change the order of the input samples. Furthermore, there is no sample rate change in the shift register so no interleaver circuitry is required at the output. The multiplexed output is passed directly to the next circuit in the echo path structure, which is the multiplexed fractional delay filter discussed in Chapter 3.

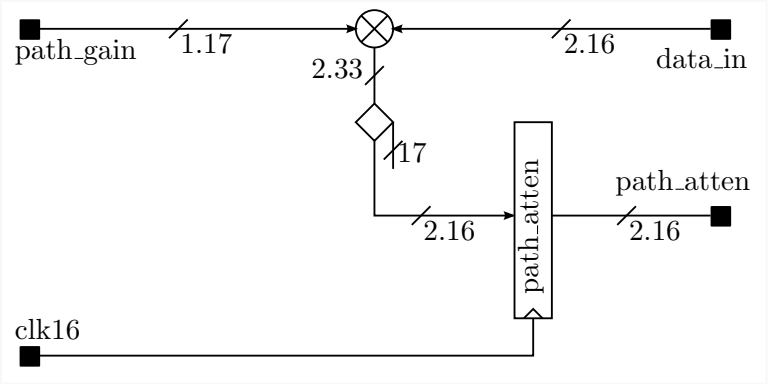


Figure A.10: Schematic for Path Attenuator

A.4.2 Path Attenuation

The path attenuation module simply consists of a multiplier followed by truncation to trim the result to 18 bits. The truncated multiplexed results are clocked out by an output register operating at 16 samples/symbol. Clocking the output helps reduce the settling time on the adder output when all echo paths are summed. This is another FIFO circuit that requires no interleaving on the output. The schematic for the path attenuator is shown in Figure A.10.

A.5 Demultiplexing Data into Parallel Streams

Before the signal is passed to the complex multiply circuit it must be demultiplexed back into parallel complex signal components. The reason for this is that the complex multiply is constructed with a pair of CORDIC circuits (discussed in Subsection A.6.3), each generating parallel sine and cosine functions. These CORDIC circuits were constructed in previous work without a multiplexed structure, and it is much easier to build a demultiplexer at the input than it is to modify the CORDIC.

The schematic for the CORDIC input demux is shown in Figure A.11. A pair of demultiplexers are used to split the input into two paths, with the real (I) component on the top path and the imaginary (Q) component on the bottom path. The stream toggle signal is used both to toggle the demultiplexers and to enable the high-rate registers running at the `clk16` rate. Low rate registers operating at the `clk8` rate are used to output the parallel

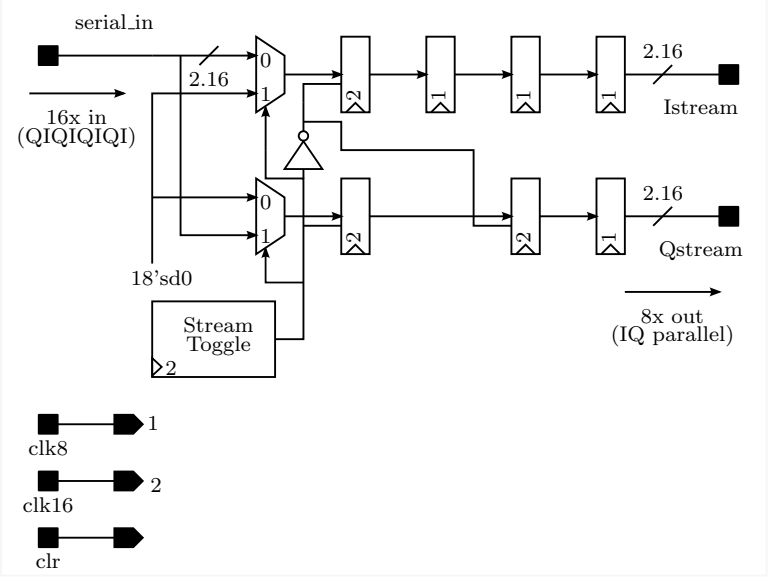


Figure A.11: Schematic for CORDIC Input Demux (Main Channel)

signals, with additional registers on the top path used to realign the timing of the data so that both components are output simultaneously.

A.6 Pipelined Successive Approximation Circuits

Communications circuits frequently require nonlinear functions such as trigonometric, logarithmic, or square root functions to perform certain signal processing tasks. In their simplest form, the algorithms that compute these functions, more often than not, require the use of one or more multiplies. As it is best to minimize multiplier use in hardware implementation, alternative algorithms must be sought. Successive approximation algorithms provide a viable alternative by using a series of additions and bit shifts in the place of explicit multiplies. FPGA manufacturers usually provide intellectual property (IP) cores with their devices which allow such algorithms to be instantiated inside the device with minimal effort. However, the rights to use these IP cores must be purchased from the device manufacturer. It is frequently less costly to construct a circuit that is specifically tailored to the task at hand.

As the name suggests, successive approximation algorithms make a sequence of guesses at a desired output value, with each guess being closer to the actual value than the previous one.

This being the case, the result is not achieved until an adequate number of approximation cycles have been performed. This implies that there is latency from input to output, as it takes several clock cycles to achieve the desired result. The desired data throughput is often too high to operate the circuit without pipeline registers between each stage, which increases latency. The pipelined successive approximation circuits used in the implementation of the channel emulator are discussed in this section.

A.6.1 Square Root

The square root algorithm chosen for this implementation is presented in [33] and is referred to by the author as the modified Dijkstra's algorithm. This relatively simple algorithm, which will not be repeated here, is advantageous not only because it is resource efficient but also because of its fast convergence time. Given an input word length of $2M$ bits, the result is obtained in M clock cycles.

The schematic for the overall square root circuit is shown in Figure A.12. The circuit begins with an initial guess of zero, depicted by the constant `36'd0` beneath the `arg` input. An input argument format with $2C$ integer bits results in an output with C integer bits. For example, an 18-bit unsigned integer input yields an output with 9.9 unsigned format.

As shown in Figure A.12, the 18-bit input argument with 4.14 unsigned format has 18 least significant bits (LSB) zero-padded to form a 36-bit unsigned input. This 36-bit precision is carried through the entire circuit. The reason for this has to do with the nature of the bit shifts, which are 2 bits to the right at each stage. An 18-bit input concatenated with 18 zero-padded LSBs will result in an 18-bit output concatenated with 18 zero-padded most significant bits (MSB). These MSBs are truncated at the output to recover the 18-bit result, which has 2.16 unsigned format.

A single stage (i.e. one approximation cycle) of the pipelined square root circuit is depicted in Figure A.13. The pipeline registers hold the intermediate results for presentation to the next stage in the circuit and are included to reduce propagation delays and increase data throughput. These registers are critical for the square root circuit as it is clocked at a rate of 160 MHz which corresponds to a clock period of only 6.25 nanoseconds.

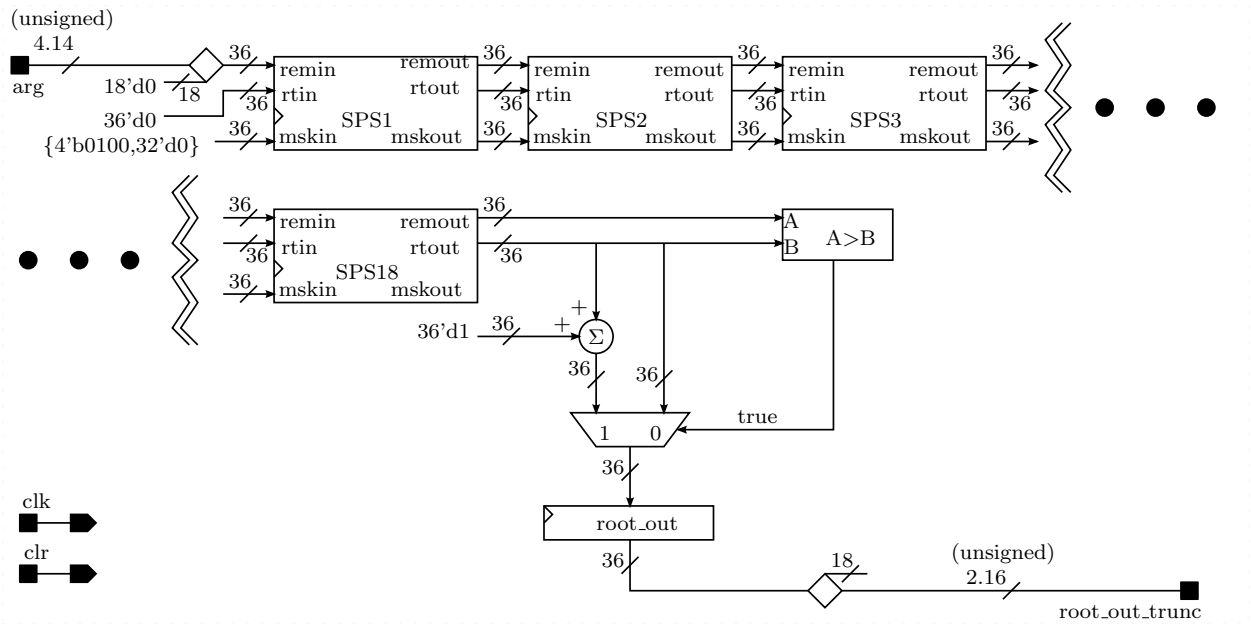


Figure A.12: Schematic for 18-Stage Pipelined Square Root

A.6.2 Natural Logarithm

The logarithm is another successive approximation circuit that can be implemented solely with a series of bit shifts and additions instead of an explicit multiplier. It is an extension of the type 1 successive approximation logarithm circuit studied and applied in EE898, a course specializing in practical hardware implementation and modern test equipment. It employs 18 pipelined stages to produce an 18-bit result using 37-bit internal carry-through precision. Although a search of the literature did not turn up any material directly related to the exact form of the algorithm that was implemented, it is a type of CORDIC implementation (see Subsection A.6.3).

At the heart of the circuit is the architecture shown in Figure A.14. The inputs to the circuit are the mantissa component `y_m` and the exponent component `y_e`, the latter of which will be discussed shortly. As the mantissa input is 1.17 unsigned format, the domain of the circuit is $\texttt{y\_m} \in [1, 2 - 2^{-17}]$. This will always produce a non-negative result. Each of the increments `dx` are provided by a LUT while the increments `dy` are implemented by applying bit shifts to the intermediate results.

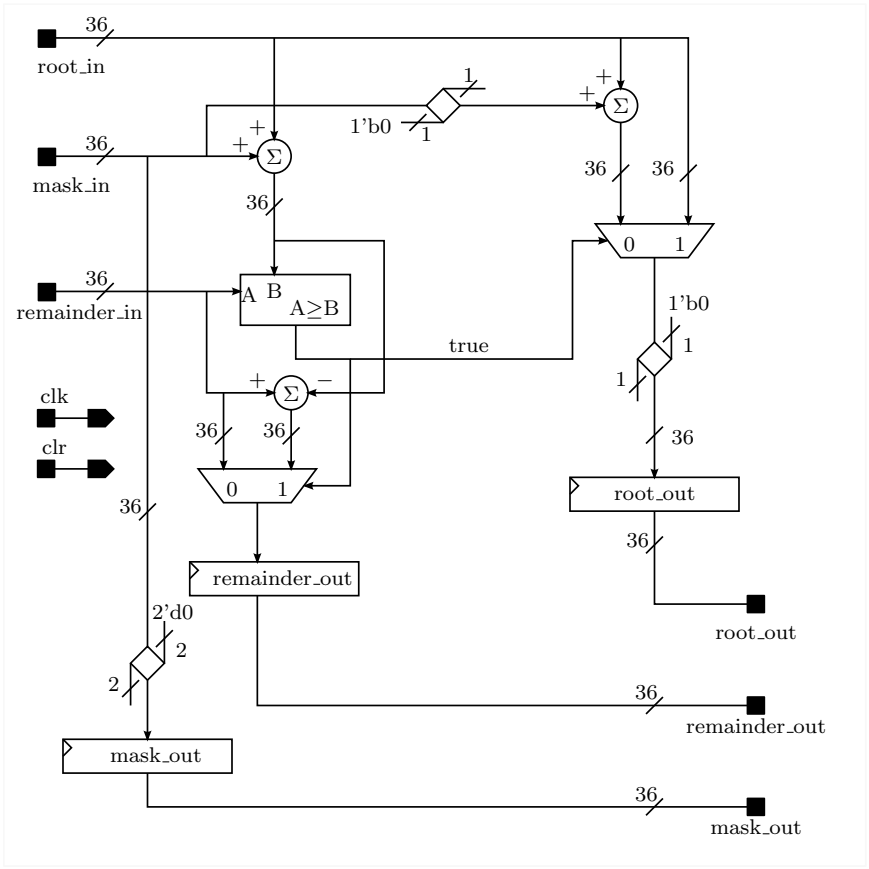


Figure A.13: Schematic for Single Stage of Pipelined Square Root (SPS)

The architecture of a single logarithm stage is shown in Figure A.15. Originally, this structure was designed to calculate a single approximation to the base-2 logarithm of the input. However, a logarithm with any base can be easily converted to a natural logarithm by using the Change of Base formula [34], which has

$$\log_a y = \frac{\ln y}{\ln a}. \tag{A.1}$$

Substitution of $a = 2$ into the formula yields a useful relationship between the natural logarithm and the hardware-friendly base-2 logarithm,

$$\ln y = \ln 2 \log_2 y, \tag{A.2}$$

which is useful for the problem at hand. Rather than applying a correction at the output,

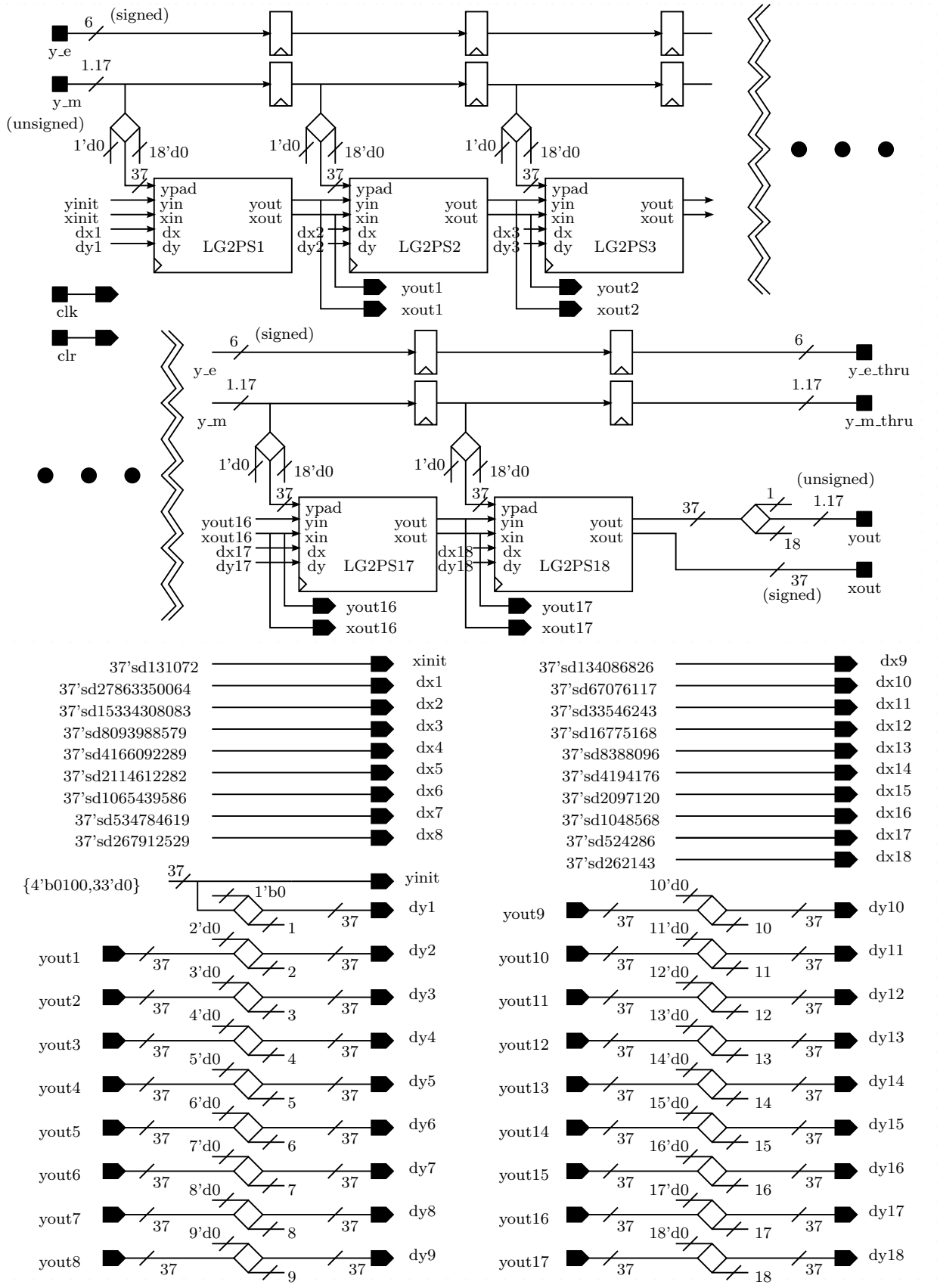


Figure A.14: Schematic for 18-Stage Pipelined Natural Logarithm

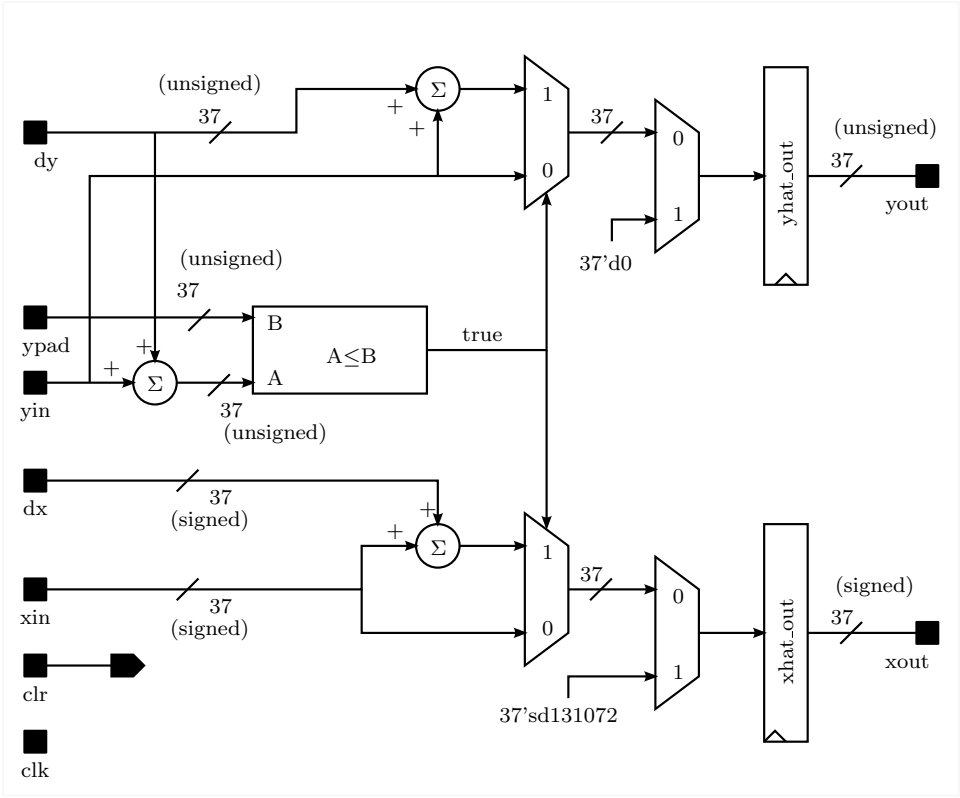


Figure A.15: Schematic for Single Stage of Pipelined Base-2 Logarithm (LG2PS)

which would require a multiplier, the constants used in the LUT are each multiplied by $\ln 2$, thus generating a set of natural logarithm constants for use in the approximations. The constant `xinit` is normally zero in the ideal case, however a correction is applied to compensate for the negative bias associated with type 1 successive approximation circuits, as they always approximate from below and never overshoot the input value.

The schematic for the entire natural logarithm assembly is shown in Figure A.16. The input `y_in` is an 18-bit unsigned fraction generated by a set of 18 independent LFSRs. As the domain of the circuit is `y_m` $\in [1, 2 - 2^{-17}]$, the preprocessor performs left bit shifts to transform `y_in` into `y_m` with the number of bits shifted being stored in the exponent `y_e` for use in the post-processing circuit.

The post-processor uses the bit shift information to apply an additive constant to `x_in` to recover the correct result. It appeals to the properties of logarithms by calculating the logarithm of a product with a sum of logarithms [35]. Specifically,

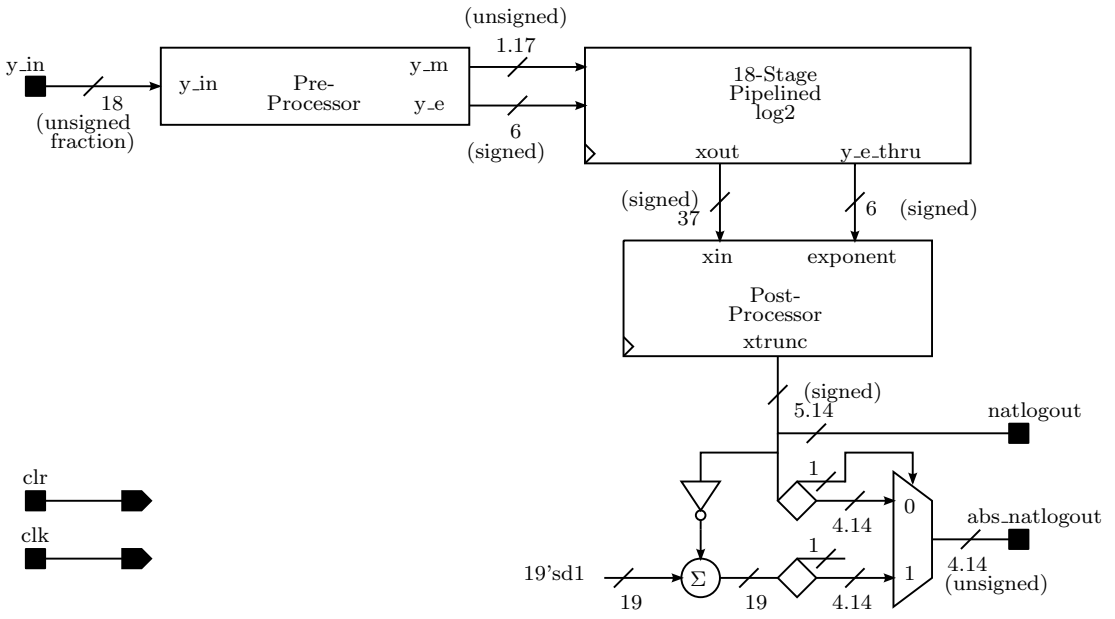


Figure A.16: Schematic for Natural Logarithm Assembly

$$\log_a(xy) = \log_a x + \log_a y, \tag{A.3}$$

which, for a natural logarithm (i.e. $a = e$), an exponent $\texttt{y\_e} = k$, and $\texttt{y\_m} = y \times 2^k$, has

$$\ln(y \times 2^k) = \ln(y \times 2^{(k+1)}) + (k-1)\ln 2. \tag{A.4}$$

Finally, as the result of the logarithm is always negative for an unsigned fraction input, another circuit is placed after the post-processor to extract the 18-bit absolute value of the natural logarithm. This is necessary to generate the random variable $f(x_1) = \sqrt{-\ln(x_1)}$ used in the AWGN generator, which is equivalent to $f(x_1) = \sqrt{|\ln(x_1)|}$ for $0 < x_1 < 1$. For the case of $\texttt{y\_in} = 0$, which mathematically corresponds to $|\ln 0| = \infty$, the hardware produces an absolute value result of $|\ln(\texttt{y\_in})| \approx 12.4767$, which is accommodated by the 4.14 unsigned format.

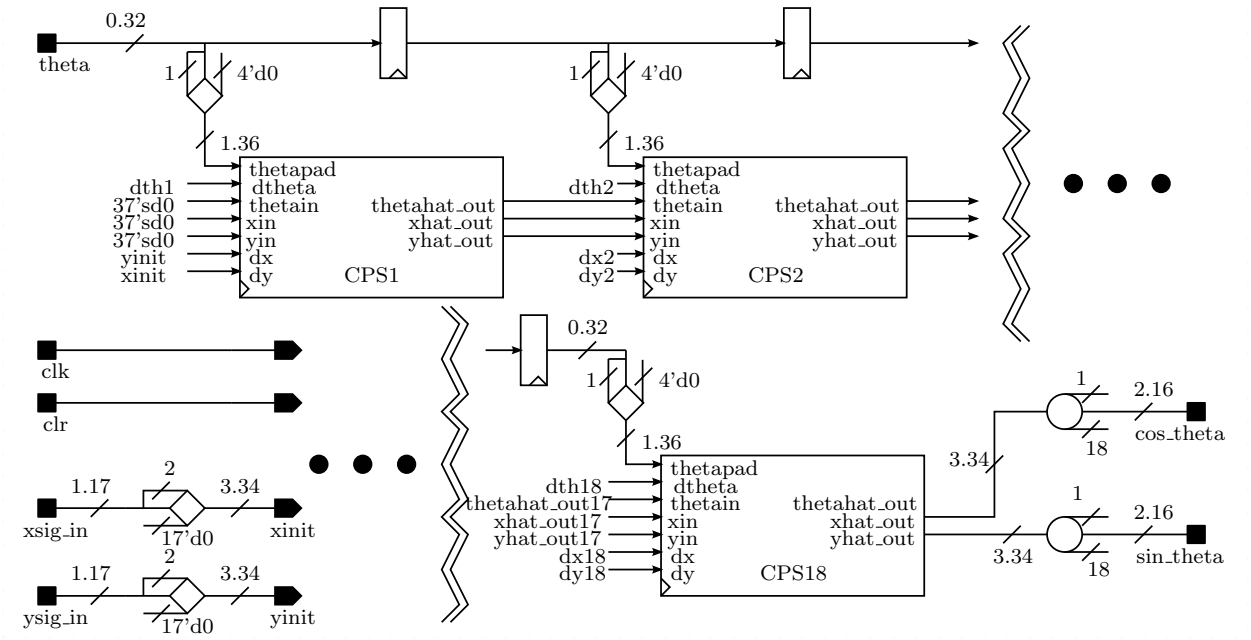


Figure A.17: Schematic for 18-Stage Pipelined CORDIC

A.6.3 CORDIC

A coordinate rotation digital computer (CORDIC) is often used in hardware to produce sine and cosine functions without the need for an explicit hardware multiplier. This type of circuit was originally developed by Jack Volder in the 1950's as a digital replacement for the analog navigation computer used in the B-58 Hustler supersonic bomber [36]. It enables trigonometric functions to be economically implemented with a series of bit shifts and additions.

The methodology for CORDIC implementation is covered in [37] and will not be repeated here. The CORDIC structure implemented in the channel emulator is shown in Figure A.17 and uses 18 stages to produce an 18-bit approximation for the sine and cosine functions. Pipeline registers are housed in each CORDIC pipeline stage (denoted by the prefix CPS), as well as the `theta` data pipeline at the top of the figure. The internal carry-through precision is 37 bits, and rounding is performed at the output of the circuit to eliminate the negative bias associated with truncation.

The architecture of the single pipeline stage is shown in Figure A.18. The angular

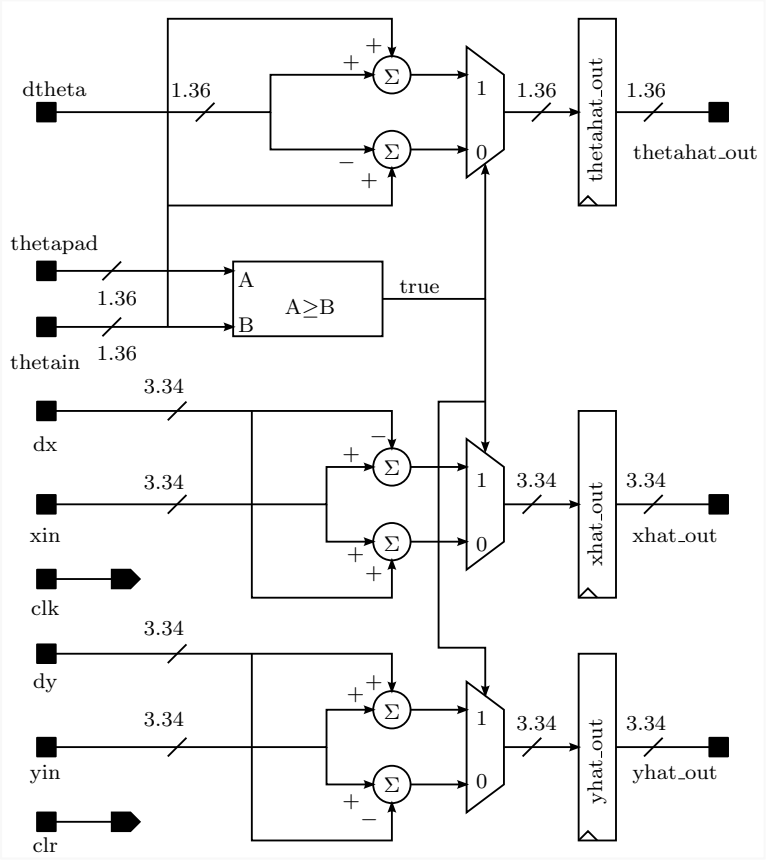


Figure A.18: Schematic for Single Stage of Pipelined CORDIC (CPS)

increment constants, `dtheta`, are provided by external LUTs (see Figure A.19). Bit shifts are also performed externally to create the `dx` and `dy` Cartesian increments. Each stage in the pipeline simply uses a comparator to determine whether or not the approximated angle `thetain` is greater than or equal to the input angle `thetapad`. The output of the comparator is then used to select the appropriate Cartesian coordinate transformation.

The schematic for the LUT constants and bit shifts is shown in Figure A.19. It is actually part of the circuit shown in Figure A.17 but is shown separately for greater clarity. The LUT constants for the angular increments are computed to 37-bit precision. Bit shifts are performed with a combination of LSB truncations and MSB sign bit appends.

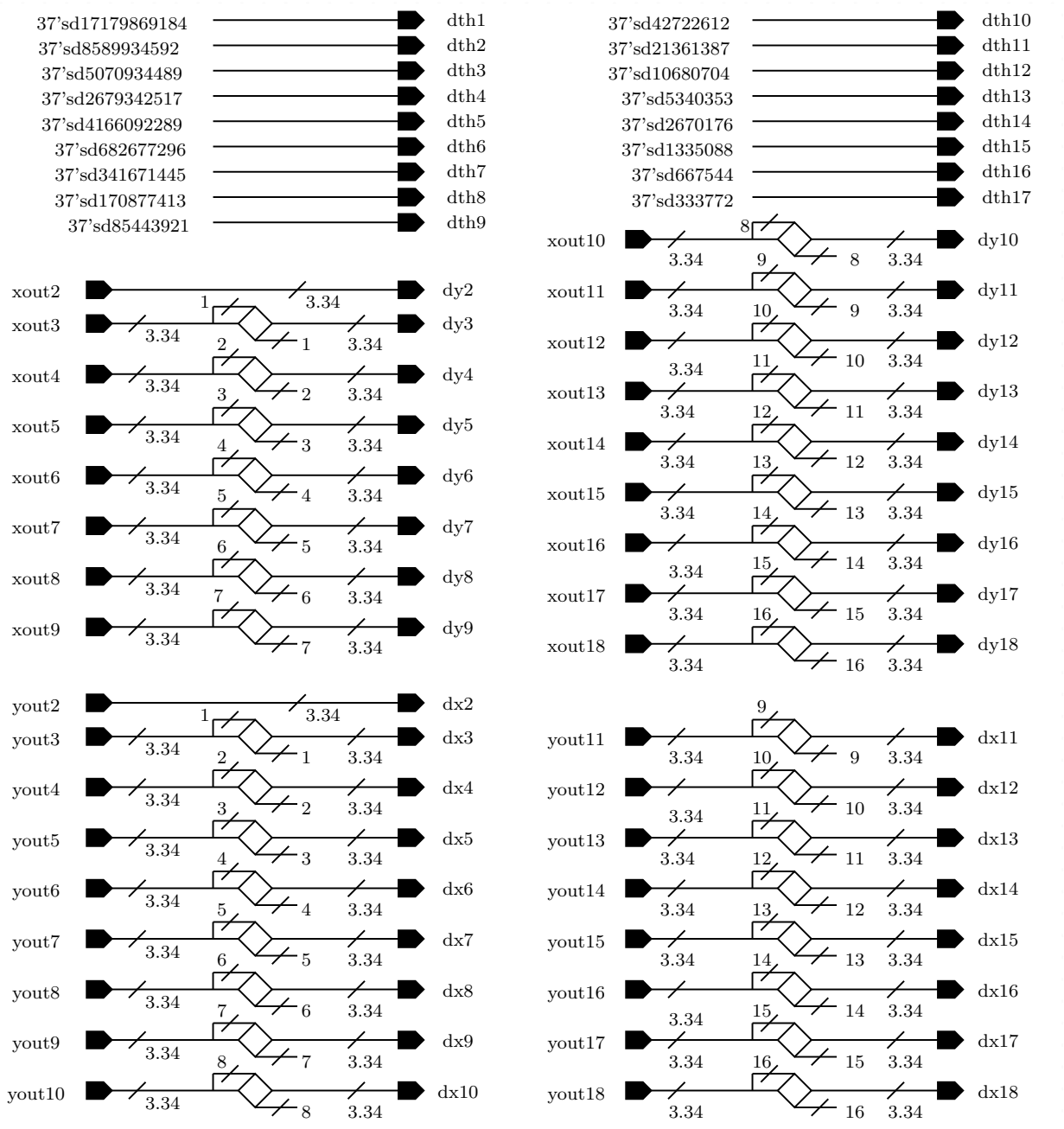


Figure A.19: Schematic for Constants and Arithmetic Shifts in 18-Stage Pipelined CORDIC

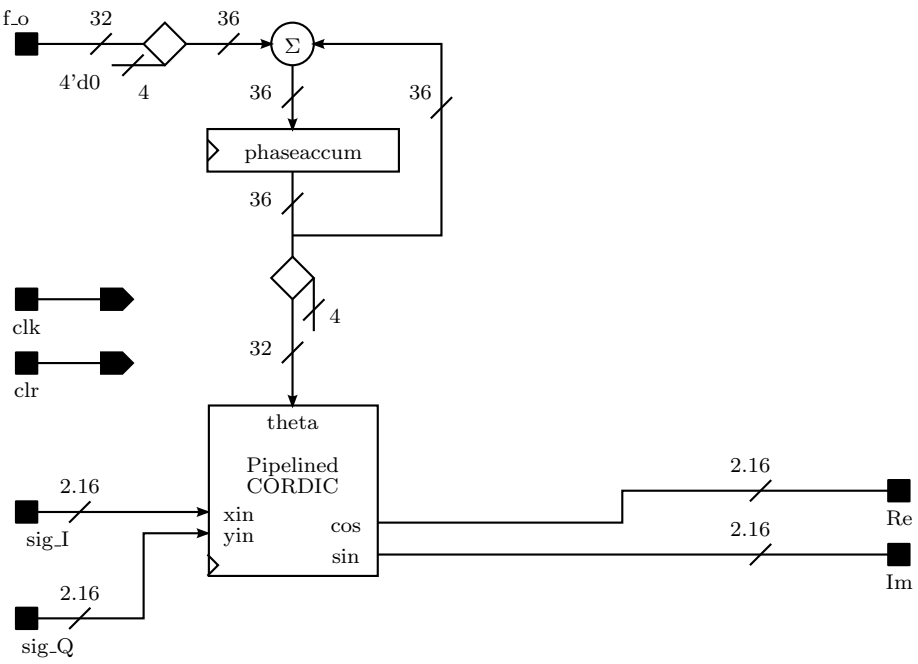


Figure A.20: Schematic for CORDIC-based Complex Multiply

A.7 CORDIC-based Complex Multiply

The architecture of the complex multiply circuit is shown in Figure A.20. The complex multiply circuit is necessary to perform frequency translation on the signals before they can be stacked. It is implemented with a pipelined CORDIC circuit, which can multiply the signal with a sine and cosine component without explicit use of a hardware multiplier. A 32-bit phase accumulator generates the phase input for each x and y (i.e. I and Q) data sample loaded into the CORDIC circuit.

A.8 Gain Boosting in Main Channel

Gain boosting in the main channel is generally less important than in the adjacent channels since the main channel is the reference. In order to maintain word precision from the output of the pulse shaping filter, its feedback gain is set to the maximum value of `17'd131071` as depicted in Figure 3.2. This output forms the 0 dB main channel reference level. The gain boosts of +6, +12, and +18 dB are implemented with data selectors and bit

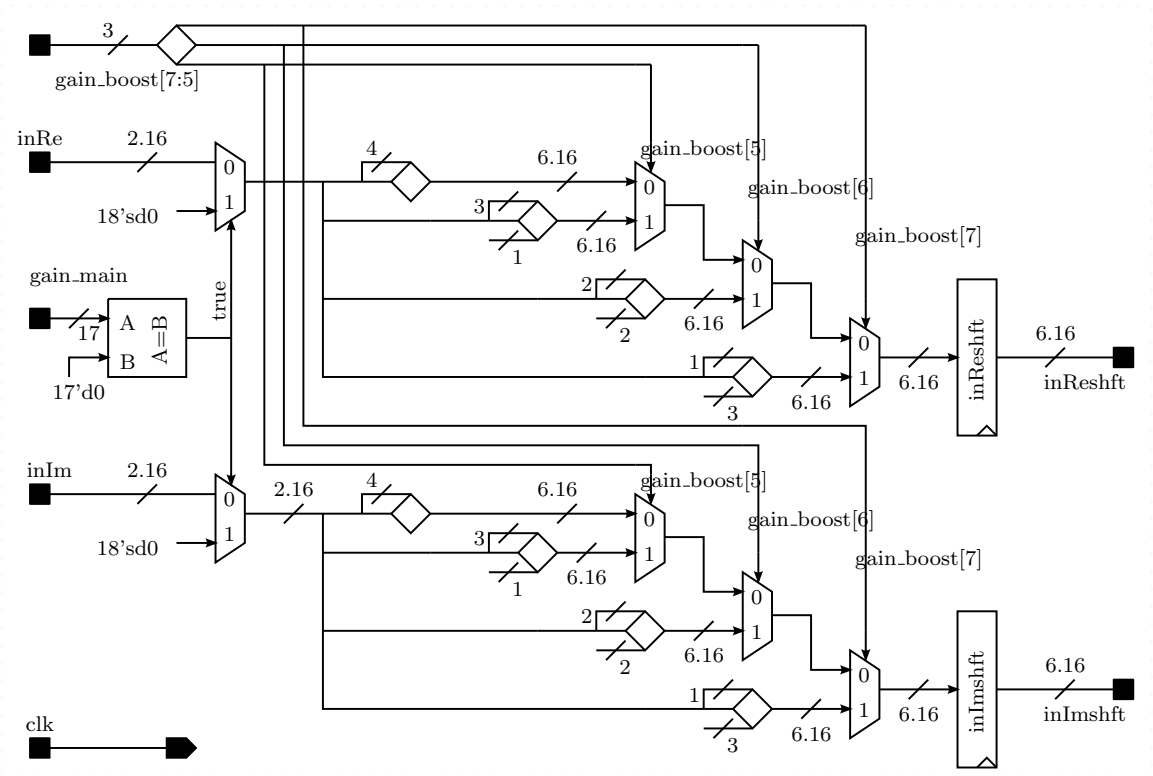


Figure A.21: Schematic for Main Channel Gain Boost

shifts so that no multiplier is necessary to apply signal gain. The schematic for the main channel gain boost is shown in Figure A.21. Note that if more than one gain boost is selected simultaneously, the order of preference is +18 dB, +12 dB, +6 dB.

A.9 Auxiliary Circuits for Adjacent Channels

Although the adjacent channel architecture was presented in Chapter 3, the internal details of each of the circuits contained therein were not addressed. Those internal details are addressed in this section.

The symbol mapper used in the adjacent channels is identical to that used in the main channel. The symbol generator differs from that used in the main channel only in that both the seed values and internal wiring of the LFSRs are unique to each channel. This is done to ensure that independent data is generated for each channel. Thus the schematics for these circuits are not repeated here.

A.9.1 Pulse Shaping in Adjacent Channels

The architecture of the adjacent channel pulse shaping filter is shown in Figure A.22. The approach to the pulse shaping architecture is identical to the main assembly in that multiple MAC circuits are used to implement time-sharing on the multipliers. What is different is that the impulse response is 16 symbols long instead of 8 symbols, necessitating the use of 8 MAC circuits. The dotted line labelled with '4 MAC Stack' in Figure 3.6 shows the structure of the 4 MAC stack blocks in Figure A.22. Furthermore, the length of the data pipeline is doubled.

As the timing of the adjacent channel pulse shaping filter remains the same, the pulse shaping filter assembly used for the adjacent channels is identical to that used in the main channel (see Figure 3.9). The same sequencer and interleaver are used.

A.9.2 Demultiplexing Data into Parallel Streams

The CORDIC input demux circuit in the adjacent channels is used for the same reason mentioned in Section A.5 — to split the complex signal back into two parallel components for frequency translation. However, due to the presence of the echo path structure in the main channel, the timing is different between the main path demux and the adjacent channel demux, although the function remains the same. The schematic for the adjacent channel CORDIC input demux is shown in Figure A.23.

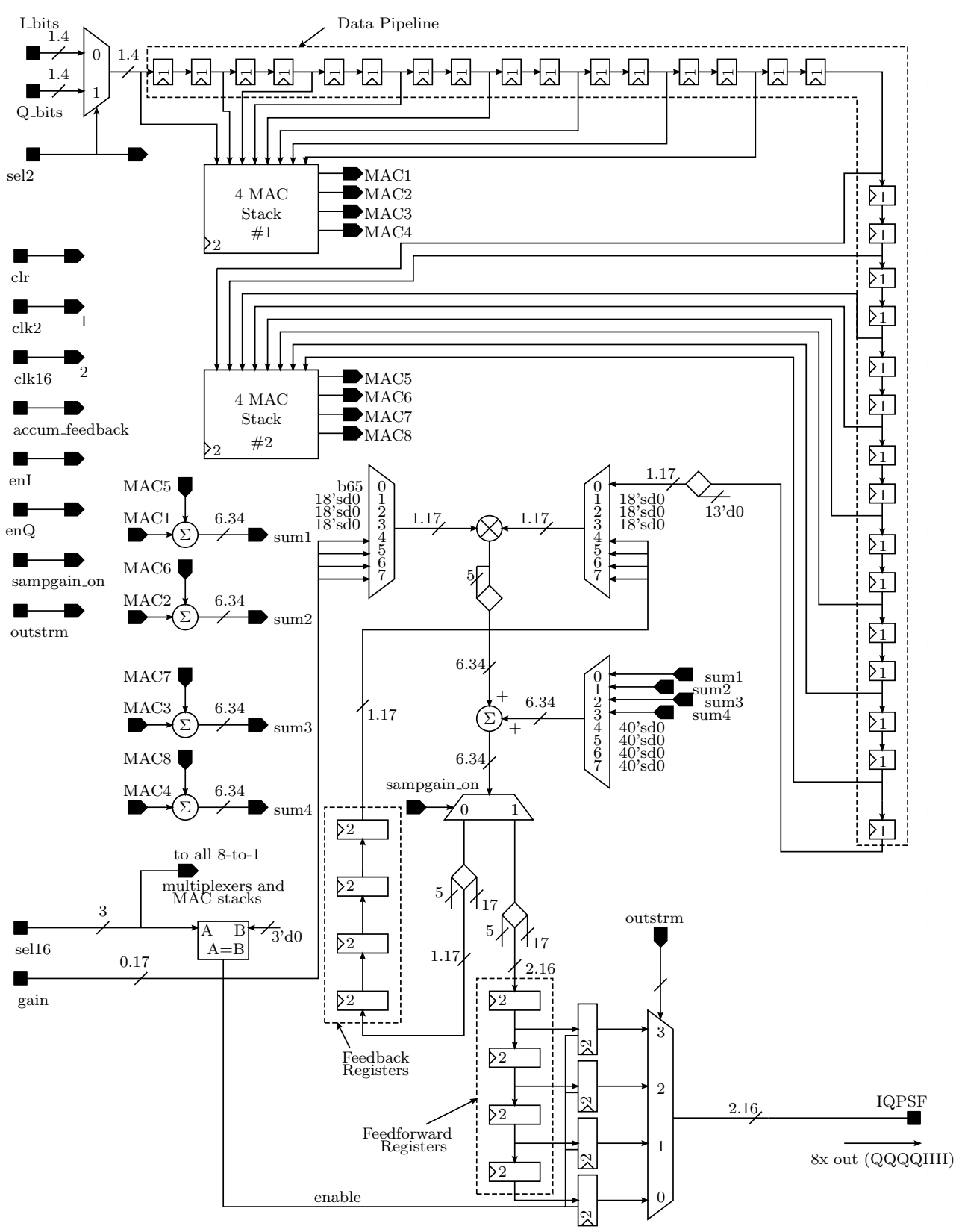


Figure A.22: Schematic for 16 Symbol SRRC Pulse Shaping Filter

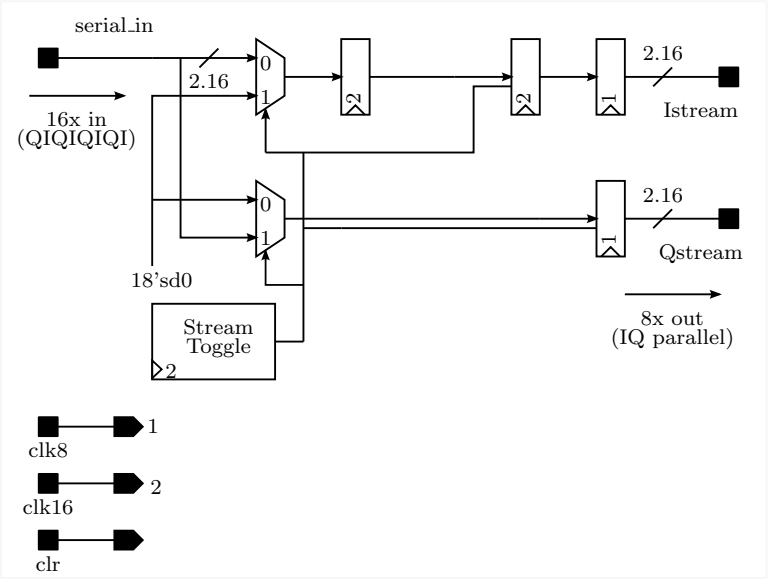


Figure A.23: Schematic for CORDIC Input Demux (Adjacent Channel)

References

[1] E. S. Smith, "The Emergence of CATV: A Look at the Evolution of a Revolution," in *Proceedings of the IEEE*, vol. 58, p. 968, July 1970.

[2] P. R. Parsons, "Two Tales of a City: John Walson, Sr., Mahanoy City, and the "Founding" of Cable TV," *Journal of Broadcasting and Electronic Media*, vol. 40, pp. 354–356, 1996.

[3] M. Mullen, "The Pre-history of Pay Cable Television: An Overview and Analysis," *Historical Journal of Film, Radio and Television*, vol. 19, no. 1, pp. 39–41, 1999.

[4] W. S. Ciciora, "An Introduction to Cable Television in the United States," *IEEE LCS Magazine*, pp. 19–20, February 1990.

[5] R. K. Jurgen, "Two-way Applications for Cable Television Systems in the '70s," *IEEE Spectrum*, pp. 46–50, November 1971.

[6] J. Gillies and R. Cailliau, *How the Web was Born: The Story of the World Wide Web*, ch. 1, p. 1. Oxford University Press, 2000.

[7] M. Haag and H. Schoof, "Telecommunications Regulation and Cable TV Infrastructures in the European Union: Current Policies and Future Issues," in *Telecommunications Policy*, vol. 18, pp. 370–372, 1994.

[8] H. Shinohara, "Broadband Access in Japan: Rapidly Growing FTTH Market," *IEEE Communications Magazine*, pp. 72–73, September 2005.

[9] M. Haag and H. Schoof, "Telecommunications Regulation and Cable TV Infrastructures in the European Union: Current Policies and Future Issues," in *Telecommunications Policy*, vol. 18, p. 367, 1994.

[10] M. Cohen-Meidan, "The Effects of Standardization Process on Competition: An Event Study of the Standardization Process in the US Cable Modem Market," in *Telecommunications Policy*, vol. 31, pp. 621–622, 2007.

[11] Cable Television Laboratories, Inc., *DOCSIS 3.0 Physical Layer Specification*, August 2013.

[12] S. Perkins and A. Gatherer, "Two-way Broadband CATV-HFC Networks: State-of-the-art and Future Trends," in *Computer Networks*, vol. 31, pp. 313–314, 1999.

[13] D. Large and J. Farmer, *Broadband Cable Access Networks: The HFC Plant*, ch. 8, p. 242. Morgan Kaufmann, 2009.

[14] D. Large and J. Farmer, *Broadband Cable Access Networks: The HFC Plant*, ch. 7, pp. 200–201. Morgan Kaufmann, 2009.

[15] D. Large and J. Farmer, *Broadband Cable Access Networks: The HFC Plant*, ch. 2, p. 19. Morgan Kaufmann, 2009.

[16] A. A. M. Saleh and R. A. Valenzuela, "A Statistical Model for Indoor Multipath Propagation," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 128–129, February 1987.

[17] J. J. Olmos, A. Gelonch, F. J. Casadevall, and G. Femenias, "Design and Implementation of a Wide-Band Real-Time Mobile Channel Emulator," in *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 749–750, May 1999.

[18] A. D. Fontaine, E. Salt, and D. E. Dodds, "A Multipath Channel Emulator Integrated With a QAM Modulator," in *26th Annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2013*, pp. 1–4, IEEE, May 2013.

[19] I. Leon W. Couch, *Digital and Analog Communication Systems*, ch. 4, pp. 231–234. Prentice-Hall, Inc., 6 ed., 2001.

[20] E. R. Bartlett, *Cable Television Handbook*, ch. 1, pp. 3–4. McGraw-Hill, 2000.

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

[22] I. Leon W. Couch, *Digital and Analog Communication Systems*, ch. 3, pp. 183–184. Prentice-Hall, Inc., 6 ed., 2001.

[23] A. W. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, ch. 7, p. 570. Pearson Higher Education, Inc., 3rd ed., 2010.

[24] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the Unit Delay: Tools for Fractional Delay Filter Design," *IEEE Signal Processing Magazine*, pp. 32–35, January 1996.

[25] L. Erup, F. M. Gardner, and R. A. Harris, "Interpolation in Digital Modems — Part II: Implementation and Performance," in *IEEE Transactions on Communications*, vol. 41, pp. 998–1008, IEEE, June 1993.

[26] J. L. Danger, A. Ghazel, E. Boutillon, and H. Laamari, "Efficient FPGA Implementation of Gaussian Noise Generator for Communication Channel Emulation," in *7th IEEE International Conference on Electronics Circuits & Systems (ICECS'2K)),Kaslik: Liban (2001)*, 2001.

[27] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, ch. 7, pp. 278–279. McGraw-Hill, 4 ed., 2002.

[28] A. W. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, ch. 7, pp. 533–538. Pearson Higher Education, Inc., 3rd ed., 2010.

[29] A. W. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, ch. 4, pp. 222–223. Pearson Higher Education, Inc., 3rd ed., 2010.

[30] M. Rice, *Digital Communications: A Discrete-Time Approach*, ch. 9, pp. 533–534. Pearson Prentice Hall, 2009.

[31] D.-U. Lee, W. Luk, J. Villasenor, and P. Y. Cheung, "A Hardware Gaussian Noise Generator for Channel Code Evaluation," in *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'03)*, IEEE, 2003.

[32] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," Application Note XAPP 052, Xilinx, July 1996.

[33] M. T. Tominska, "An Area-Efficient Implementation of a Fast Square Root Algorithm," in *Proceedings of the 2000 Third IEEE International Caracas Conference on Devices, Circuits and Systems*, pp. S18/1 – S18/4, IEEE, March 2000.

[34] J. Stewart, *Calculus: Early Trancendentals Single Variable*, ch. 1, p. 69. Thomson Brooks/Cole, 5 ed., 2003.

[35] J. Stewart, *Calculus: Early Trancendentals Single Variable*, ch. 1, p. 68. Thomson Brooks/Cole, 5 ed., 2003.

[36] J. E. Volder, "The Birth of CORDIC," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 25, pp. 101–105, June 2000.

[37] J. E. Volder, "The CORDIC Trigonometric Computing Technique," in *IRE Transactions on Electronic Computers*, vol. EC-8, pp. 330–334, 1959.