SPACE COMMUNICATION CHANNEL EMULATION USING DIGITAL AND ANALOG SIGNAL PROCESSING

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Tom Hickok

March 2010

COMMITTEE MEMBERSHIP

TITLE:	Space Communication Channel Emulation using Digital and Analog Signal Processing
AUTHOR:	Tom Hickok
DATE SUBMITTED:	March 2010
COMMITTEE CHAIR:	Albert A. Liddicoat, Ph.D.
COMMITTEE MEMBER:	James Harris, Ph.D.
COMMITTEE MEMBER:	Fred DePiero, Ph.D.

Abstract

Space Communication Channel Emulation using Digital and Analog Signal Processing

Tom Hickok

New communication protocols intended for large distances, including low orbit and deep space, can be inherently difficult to evaluate since trial implementations are often impractical. In order to accurately measure the performance of a new protocol, it is important to evaluate it in an environment that most closely matches that in which it will be used. This thesis demonstrates the ability to emulate a space communications channel through digitizing a transmission centered at an intermediate frequency of 70 MHz with a bandwidth of 24 MHz, digitally introducing the characteristics of a transmission through space, and reconstructing the digital data to its analog counterpart. Delay, Doppler shift, Gaussian noise, and fading are among the most prevalent characteristics of such a channel, and thus were the focus of this thesis. Special care was given to the design of each digital and analog component to maintain the integrity of the original signal by minimizing all undesired noise introduced. The final design can accurately produce a given dynamic transmission signature or continually output a static set of channel characteristic parameters to test new communication protocols.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Testing space communication systems and associated protocols is a daunting task when considering the operating conditions in which these systems will be utilized. Often, such tests can truly only be performed directly by putting the systems in their respective operating environments, in the vastness of space and moving at high velocities, which not only proves to be impractical, but also eliminates the ability to change and improve the systems in operation. Emulating the distances and velocities of these systems as they will be used allows for laboratory testing which enables engineers to make necessary adjustments and improvements before final implementation. The scope of this research project is the design of such an emulator to reproduce the operating environment of these systems as accurately as possible.

The environment these systems will be subject to is referred to as the space communication channel or RF channel. The term "channel" is used here in a broad sense. The most general sense of the word can be described as everything between the source and the destination of a signal. A common communication scheme is shown in Figure 1.1. Here, the domain of the channel, as used in this work, can be seen starting from the intermediate frequency (IF) signal of the transmitter to the IF signal of the receiver. Thus, the emulator must accurately reproduce the effects of everything in between these two points.

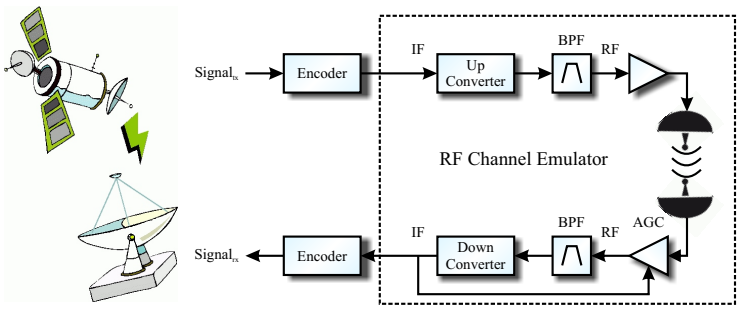


Figure 1.1: A Typical Communication Scheme. The domain of the channel, as defined in this work, is depicted.

In essence, the emulation of an RF channel may be thought of as a representation, in mathematical or algorithmic form, for the transfer characteristics of any contiguous subset of the communication medium. [12] A channel model may include any number of physical medium elements including waveguides, wires, the atmosphere, or the ideal medium, free space. Although generally these elements are modeled to fit external observations, much of this work is based around the underlying physical phenomena experienced through the relative distance and velocity of the transmitter and receiver.

The RF Channel Emulator is designed to emulate different communication scenarios for NASA's future space missions. A relatively simple scenario is to create the dynamic characteristics of the RF channel between a spacecraft and a ground station on the moon. In this scenario, the spacecraft passes above the ground station at an orbital altitude of approximately 300 km for a period of several minutes. During this time, the delay changes slightly with the relative distance between the spacecraft and station which creates a small and varying Doppler Effect, the signal is dynamically attenuated depending on the relative orientation and geometry of the antennae, and noise is dynamically varied depending on the necessary gain required to accommodate the attenuation.

Another possible scenario is emulating communication between two spacecraft. Here, one of the spacecraft might be near the earth while the other orbits

the moon. This will result in a propagation delay of 1-2 seconds. The effects of the physical phenomena that alter the signal in these dynamic scenarios will be created by the RF Channel Emulator.

This thesis presents the implementation and verification of the RF Channel Emulator and its ability to reproduce typical dynamic communication scenarios as described above as well as static and semi-static communication conditions within a scenario. The RF Channel Emulator is a mixed analog and digital system. Many of the analog portions of the system were designed and tested by Kyle Woolrich [22], although some of these portions are revisited in this work. The entirety of the digital portions are covered in this thesis including design considerations, system development, and the test and measurement for the digital portions as well as refinements to the analog portions of the RF Channel Emulator.

The first few chapters provide a higher level view of the elements required in the implementation of the RF Channel Emulator. Chapter two of this thesis gives a background of the physical phenomena common in a space communication channel. This includes discussions of the Doppler Effect, delay, additive white Gaussian noise, and fading. Chapter three contains a detailed description of the overall system architecture. Furthermore, this chapter includes a background into the theory required for design choices of the various digital components as well as modifications of some of the analog portions previously done by Woolrich [22]. Chapter four provides a system level comparison of the RF Channel Emulator with similar systems on the market.

The remaining chapters provide a more detailed analysis of the system development, component verification, and integration of the RF Channel Emulator. Chapter five is a comprehensive verification of each digital component in the system as well as the major analog and mixed signal components. Chapter six discusses the integration and final system functionality including the user interface. Finally, chapter seven provides a brief conclusion, possible future work that

can be pursued, and a description of contributions made by this work.

Chapter 2

Background

Before discussing the design and characterization of the RF Channel Emulator, this chapter will provide some of the essential definitions and theory behind the major operating principals. As mentioned in the introduction, the primary features of the emulator are modeled to fit the underlying physical phenomena of the channel. The main phenomena–the Doppler Effect, additive white Gaussian noise, fading, and delay–are described here to assist in the understanding of their respective model implementations in the chapters to follow.

2.1 Doppler Effect

The Doppler Effect is a physical phenomenon that occurs when a signal is transmitted from a source to a receiver with a relative non-zero velocity between the two. This velocity creates a perceived shift in the frequencies of sinusoidal signals transmitted that is proportional to the frequency and dependent on the relative velocity. This effect goes hand-in-hand with delay as well since the relative velocity between the source and the receiver describes the rate of change in the distance between the two.

Observations of the Doppler Effect are not limited to the electromagnetic spectrum in which radio frequencies reside. Any transmitted signal that has a

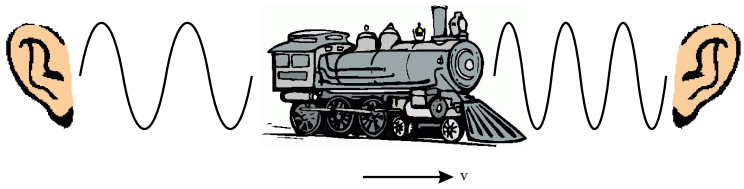


Figure 2.1: A Common Example of the Doppler Effect

known propagation velocity will be influenced by this phenomenon. The pitch (frequency) of a train whistle approaching is quite easily discernible as a higher frequency than when it is leaving. This occurs because, to the observer, the propagation velocity of sound through the air remains constant while the source, the train whistle, is moving relative to both the air and the observer. The whistle creates periodic high and low points of pressure in the surrounding air; the distance between these points is closer in front of the train when the train moves forward but are further apart behind. The frequency shift can be expressed in terms of the velocity (v_S) of the source, in this case the train, and the velocity in which the signal travels (v), in this case the velocity of sound through the air. Since the velocity of sound through the air changes relative to the velocity of the listener, the listener's velocity (v_L) must also be taken into account. The frequency at the listener (f_L) can be expressed as follows relative to the frequency at the source (f_S).

$$f_L = \frac{v + v_L}{v + v_S} f_S \tag{2.1}$$

The effect works essentially identical in the electromagnetic domain with minor exceptions. Einstein noted that since the propagation velocity of electromagnetic signals appear to be constant regardless of the point of view, the relative time of the moving party, be it the source or the receiver, appears to slow down from the other's point of view. Thus, not only are the waves themselves compressed or expanded as in the example with the train whistle, but the frequency

at which they appear to be transmitted slows due to the apparent time dilation. The relativistic Doppler Effect in terms of frequency can be expressed as follows:

$$f_{rx} = \sqrt{\frac{1 \pm \frac{v}{c}}{1 \mp \frac{v}{c}}} f_{tx} \tag{2.2}$$

Here, f_{tx} is the transmitted frequency from the point of view of the source, while f_{rx} is the observed frequency at the receiver. The constant c is the speed of light, and v is the relative speed of the source toward the receiver.

Since the application of the RF Channel Emulator is to reproduce the effects of traveling spacecraft, $|v|$ is going to be much smaller than c. This can simplify the equation down to the following relationship:

$$f_{rx} \approx f_{tx}(1 \pm \frac{v}{c}) \tag{2.3}$$

It is important to note that f_{rx} is dependent not only on c and v but on f_{tx} as well. This means that the the transmitted frequency is not simply shifted to reproduce what is received but rather is multiplied by a scaling factor, $(1 \pm \frac{v}{c})$. Scaling the signal in the frequency domain becomes necessary to accurately reproduce the Doppler Effect rather than simply performing a frequency shift. An example of this can be seen in figure 2.2.

$$\Delta f \approx \frac{v}{c} f_0 \tag{2.4}$$

The Doppler Effect is a change in frequency as perceived by the receiver, from the nominally transmitted frequency, caused by the relative velocity of the source and the receiver [20]. The various frequencies in the transmitted signal must be accounted for in order to improve the performance of a communication system. A simple solution would be to know the range or band a carrier frequency is expected to be received and simply band-pass all frequencies within that range.

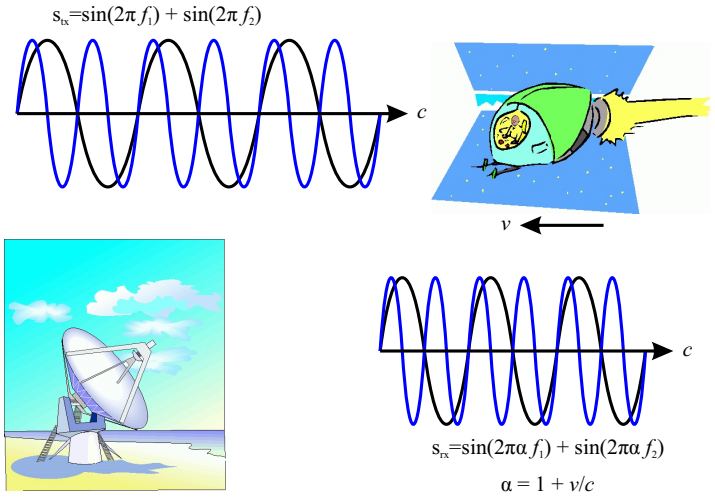


Figure 2.2: Example of Frequency Scaling due to the Doppler Effect. Δf increases with f_0.

The consequence of this approach is that there is much noise energy passed which will result in a poorer signal-to-noise ratio (SNR). Alternatively, better performance is achieved by the use of a carrier-frequency tracking loop which allows a narrower frequency band to be passed improving the SNR.

2.2 Delay

The propagation delay of a transmitted signal is always non-zero between two distinct points given a finite propagation velocity. For much of terrestrial communication, the distances in proportion to the speed of light (propagation velocity) are such that delays are negligible. With space communications, the distances are vast enough to incur significant delays. Space communications, as discussed in the previous section, also have a high enough velocity that the Doppler Effect must be taken into account. By definition, velocity is the rate of change of distance and thus the rate of change of delay. This dynamic delay must also be incorporated into an emulator with the goal of accurately reproducing the

effects of space communications.

Communication systems that deal with long delays must have protocols in place that can tolerate the long delays and still transfer data quickly and accurately. As an example, the average distance between the earth and the moon is roughly 384,303 km. Traveling at the speed of light, propagation times are roughly 1.3 seconds one-way. This makes a round trip transmission roughly 2.6 seconds. Terrestrial signals, on the other hand, rarely experience delays of this magnitude from node to node.

2.3 Noise

Noise is defined as the undesired electrical signals in a given system. The presence of noise inhibits the receiver's ability to correctly distinguish the signal's information [20]. Noise comes from a variety of sources both man-made and natural. Natural noise includes atmospheric disturbances, radiation from galactic sources, and noise inherent in the electronic hardware. The dominant source of noise is usually that which is inherent in the hardware itself.

The typical model for the noise inherent in the hardware is white Gaussian noise. White noise has a power spectral density that is constant across all frequencies. Pure white noise can't realistically be used given that the spectral range is infinite which would require an infinite power source to produce. Therefore, the noise is often bandwidth limited to something much greater than the bandwidth of interest in a system such that a finite bandwidth can be used to approximate noise with infinite bandwidth [20].

A common approach to introducing noise to a system for analysis is to simply add the noise to the system. This is referred to as the Additive White Gaussian Noise (AWGN) model. The AWGN channel is a good model for many satellite and deep space communication applications given that this is the model best suited

to emulate the noise introduced by the inherent noise in the electronic system itself. Terrestrial communications, however, have to deal with other phenomena such as multipath, terrain blocking, and interference.

2.4 Fading

All forms of communications must deal with the limitations and inherent difficulties imposed by transmitting through the medium. The medium is often a dynamic system changing over time as is often the case with wireless communications. For example, in satellite communications, the weather can have a significant impact on the propagation characteristics of the signal through the atmospheric medium. The effect of meteorological alterations causes the attenuation of the signal to change over time [3]. Antenna patterns and the relative geometry between the source and receiver can cause more deterministic but similar changes in the attenuation of the signal.

Having distinctly different paths in which a signal could travel from the source to the receiver, simply called multipath, can also affect the attenuation of the received signal in a variety of ways (see figure 2.3). The propagation times for each path can create constructive and destructive interference dependent on the difference in time and frequencies transmitted. The inherent propagation characteristics of each path will also contribute to the overall multipath effect as can be seen in figure 2.3.

There are two popular statistical models used to determine the effects that fading may have on the reliability of a communication system. First, Rician fading is a non-zero mean stochastic model that describes the fluctuations in the received signal's amplitude. This fading model is commonly used when the line-of-sight signal dominates any multipath signals. When no one path dominates the others, the Rayleigh fading model is often used [19]. Rayleigh fading is a zero-mean stochastic model that describes the fluctuations in the received signal

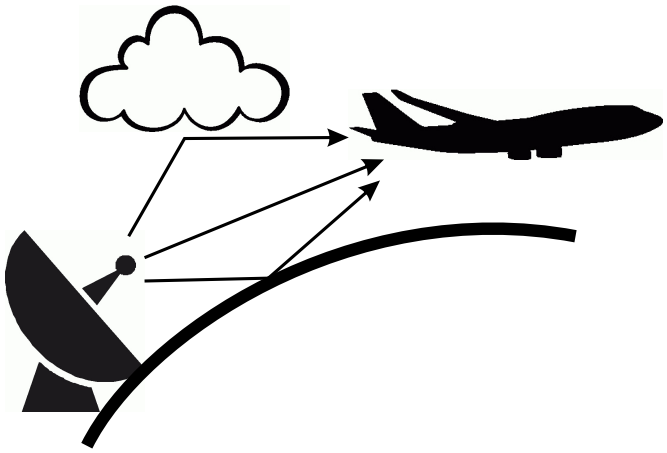


Figure 2.3: Diagram of a Multipath Communication Channel. The transmitted signal reaches the receiver following multiple paths of varying path lengths. These varying lengths cause interference to the signal at the receiver. This figure illustrates a line-of-sight path, which is characteristic of Rician fading.

amplitude. In space communications, the line-of-sight signal is almost exclusively the dominant signal. Therefore, the Rician fading model is a more accurate representation in these circumstances.

Chapter 3

System Description

This chapter gives an overall implementation approach for both the analog and digital portions of the RF Channel Emulator. The system requirements, included in Appendix A, are used to outline design considerations and component or topology choices. Various design options are considered and discussed for some analog portions and all digital portions of the system.

3.1 General Description

The RF Channel Emulator uses a combination of both analog and digital techniques to accurately emulate the space link channel. The space link channel, in this case, is the RF signal path from the transmitter, while the signal is at the IF stage, to the receiver's IF stage. Emulation of this path will introduce the effects of the physical phenomena that are inherent to it, namely the Doppler Effect, delay, noise, and fading. This will allow the emulator to be placed between the communication systems under test for a realistic analysis. The emulator is digitally controlled from an external computer through an Ethernet connection. Figure 3.1 displays an example test setup in a laboratory setting.

The emulator works at the IF stages of both the source and receiver since digitizing actual transmission frequencies is impractical. As dictated by the sys-

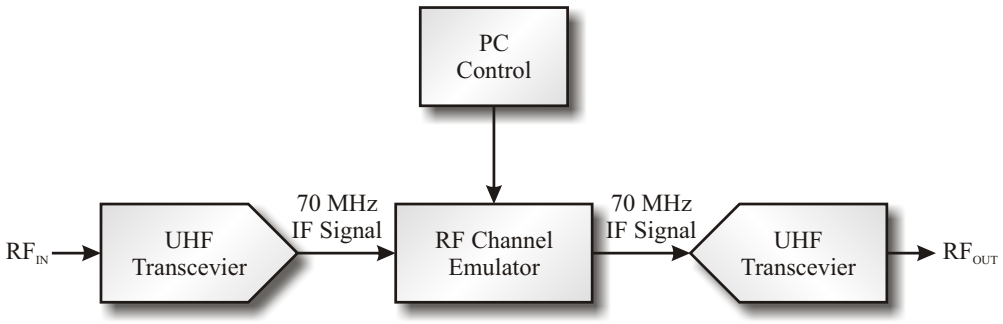


Figure 3.1: RF Channel Emulator Laboratory Setup [Appendix A].

tem requirements provided by JPL (Appendix A), the emulator uses a 70MHz Intermediate Frequency for the channel emulation. The IF signal is used to introduce the effects of the channel. The following sections of this chapter discuss the requirements, system implementation and design.

3.2 System Requirements and Overview

The RF Channel Emulator contains both analog and digital components. An overall implementation block digram is shown in figure 3.2. The emulator consists of five major sections at the top level. First, there is the analog input stage responsible for the accurate digitization of the incoming IF signal. Next, there is the digital path, implemented in an FPGA, that introduces the emulated effects of the space link channel. The signal is then reconstructed to analog form through the analog output stage. The DDS analog section is responsible for creating the clock signals necessary for the three previously mentioned sections to operate correctly. Finally, the control software resides on an external computer and is used to control the channel's parameters to produce different communication scenarios.

A summary of the requirements for each of the five components can be seen in table 3.1 as defined in the statement of work provided by JPL(Appendix A).

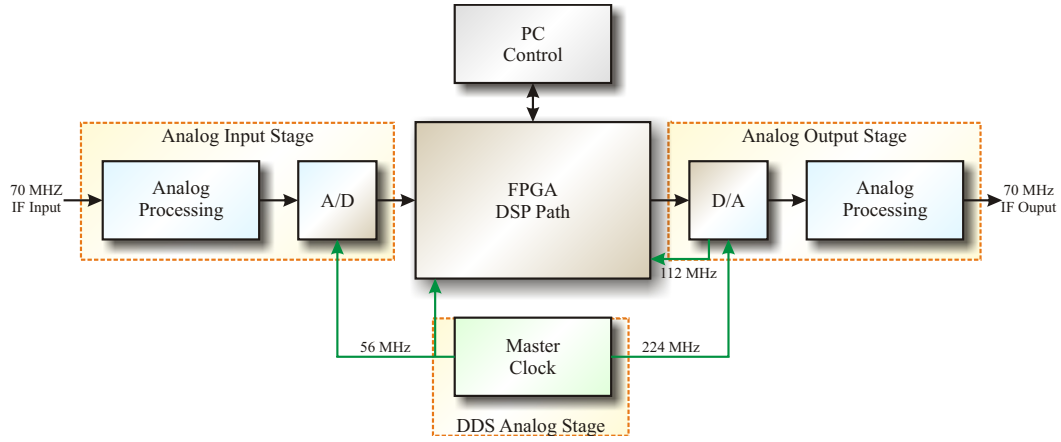


Figure 3.2: Overall System Block Diagram of RF Channel Emulator. [Discussion of top level diagram].

RF Channel Emulator Section	Requirement	
Analog Input Stage	Input Power:	-7 dBm to -12 dBm
	Center Frequency:	70 MHz
	Bandwidth:	24 MHz
Analog Output Stage	Output Power:	-12 dBm(max)
	Center Frequency:	70 MHz
	Bandwidth:	24 MHz
FPGA DSP Path	Delay:	1 msec to 2.0 sec
	Noise:	Minimum of 4 standard deviations. Independently applied to I and Q channels.
	Attenuation:	0 dB to 50 dB. Independently applied to I and Q channels.
	Doppler Shift:	Uniform frequency shift: $\pm$3.5 kHz Resolution: 1.0 H Instantaneously Rate: 1000 Hz/sec to 200 Hz/sec
External PC Control	Interface:	Ethernet
	GUI:	Controls system parameters
Energy Per Bit to Noise Desity Ratio	E_b/N_0	10 dB Maxiumum 0 dB Minimum

Table 3.1: RF Channel Emulator System Requirements. Appendix A.

3.3 Analog Input Stage

The analog input stage receives the signal from the UHF transceiver and conditions it to be digitized minimizing any aliasing in the process. Figure 3.3 is a block diagram for the implementation of the analog input stage [22].

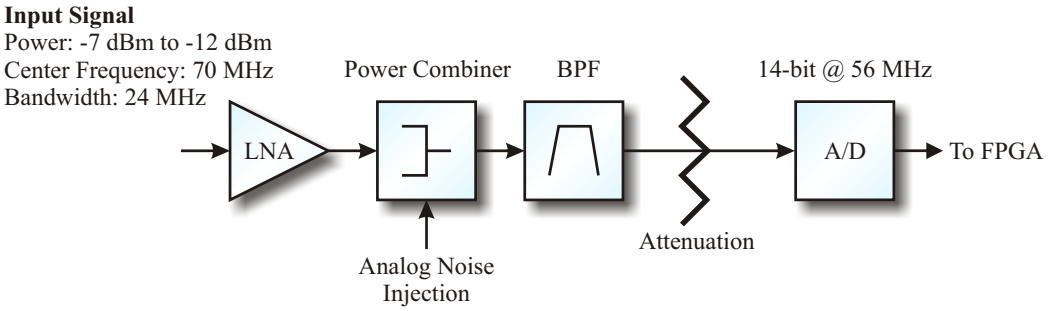


Figure 3.3: Block Diagram of the Analog Input Stage [22].

The UHF transceiver sends a signal with a center frequency of 70 MHz and a bandwidth of 24 MHz to the RF Channel Emulator. The first component of the input stage is a low-noise amplifier. The output of this is fed into a power combiner which allows an outside analog noise signal to be added to the original signal in the case that a noise signature other than the Gaussian white noise created in the FPGA is desired. After the power combiner, the signal is routed through a bandpass filter (BPF) and attenuated prior to being digitized by the analog-to-digital converter (A/D). The A/D undersamples the signal at 56 MHz and sends the digitized signal to the FPGA. Undersamping has the effect of down-converting the original analog signal centered at 70 MHz to a digital signal centered at 14 MHz.

There are two primary design considerations for the analog input stage. The first is to minimally contribute any unwanted noise to the signal during the digitizing process. The second is to assure that only the band of interest is digitized to greatly reduce any aliasing that might take place from the down conversion. The input analog stage of the RF Channel Emulator can negatively affect the

signal-to-noise ratio of the incoming signal with the inherent noise in each component [11].

The signal-to-noise ratio degradation can be quantitatively measured by the noise figure. Each component has a noise figure associated to it which can be combined with the other components to create a composite noise figure. The composite noise figure can be described with the following formula with n components.

$$F_{comp} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + ... + \frac{F_n - 1}{G_1 G_2 ... G_{n-1}} \tag{3.1}$$

It can easily be seen that the first component in a series predominantly determines the noise figure of the entire series. Ideally, the first stage in a series should have a low noise figure and a high gain to minimize the noise figure contributions of the subsequent components.

Aliasing of unwanted signals outside the band of interest is also a large concern in minimizing SNR degradation. When sampling, any frequency greater than one-half the sampling frequency (f_{sample}) will be aliased to a frequency within that range (0 Hz to $f_{sample}/2$). It is this fact that makes undersampling possible to down-convert the IF from 70 MHz down to 14 MHz with a 56 MHz sampling frequency, but it is also this undersampling that will create a noise component in the digital domain due to any signals outside the band of interest in the analog domain that are not properly attenuated [10]. An example of this is shown in figure 3.4 where there is noise outside of the band of interest. Figure 3.5 shows the digital frequency spectrum that results after sampling at f_{sample}.

Once the signal is digitized, it is impossible for the out-of-band noise to be distinguished from the desired signal. Thus, the analog input stage must attenuate any signals out of the band of interest. An example of this is shown in figure 3.6 and figure 3.7. The former figure shows the input signal in the analog frequency domain with the passband of the bandpass filter superimposed. The

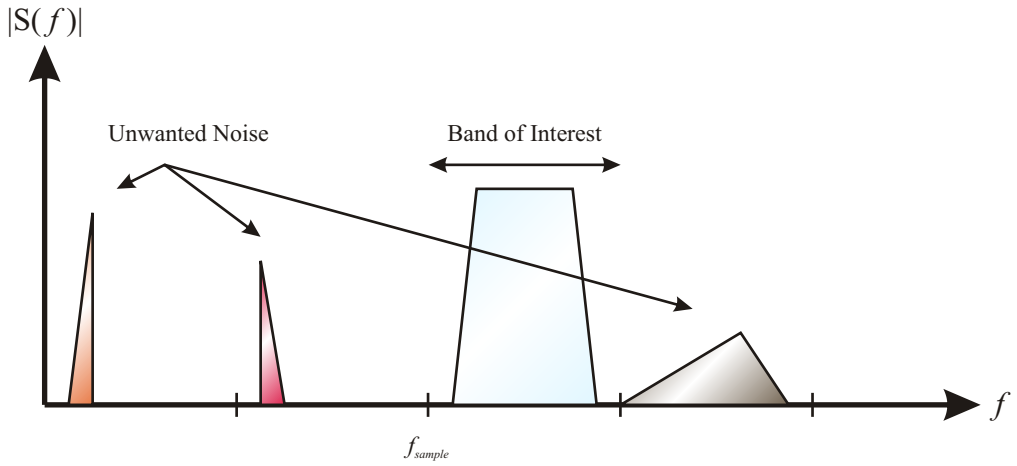


Figure 3.4: Possible Noisy Input Signal. A possible input signal in the analog frequency domain with noise outside of the band of interest.

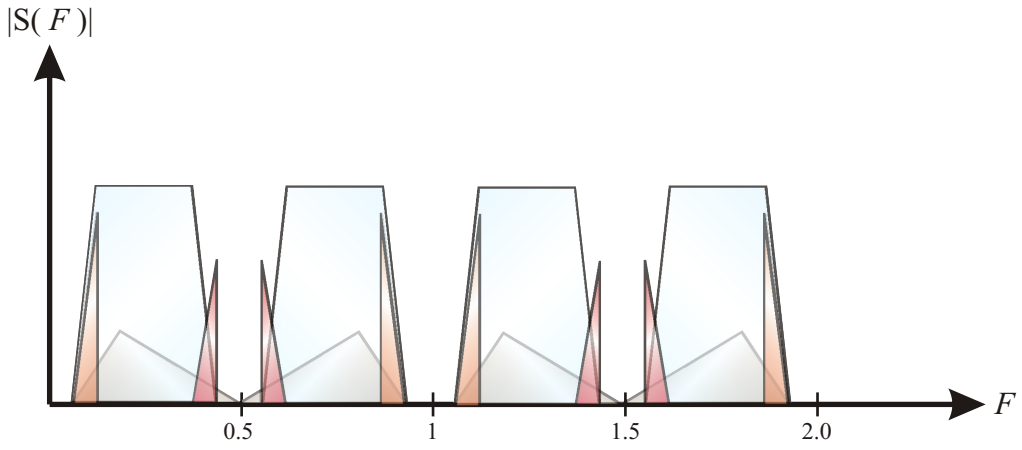


Figure 3.5: Aliasing Effects of a Sampled Noisy Signal. Without any attenuation of the out-of-band noise, aliasing takes place corrupting the signal in the band of interest.

latter shows the digital frequency response after sampling at the output of the bandpass filter with a minimal aliasing effect.

The attenuation of the signals outside the band of interest is done through the use of a Surface Acoustic Wave (SAW) filter. SAW filters in general have relatively flat responses in the passband while having sharp transitions to out-of-

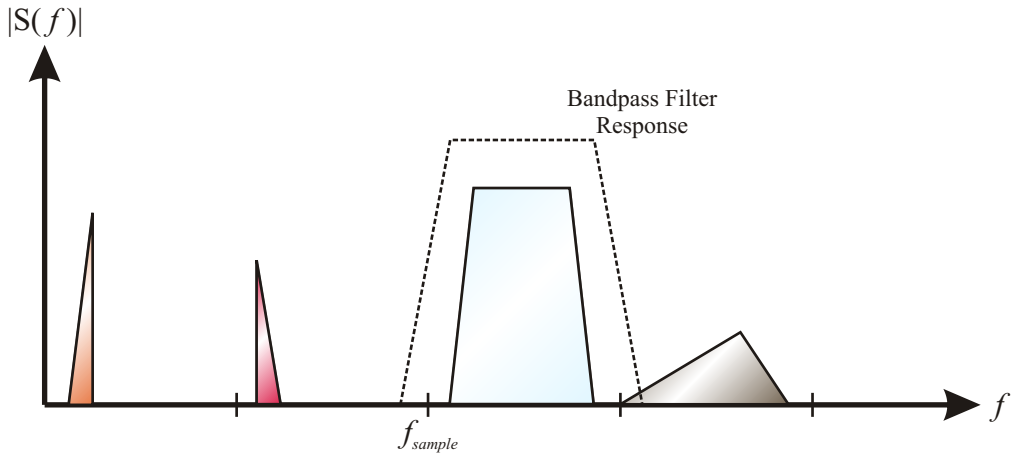


Figure 3.6: Possible Noisy Input Signal with BPF Response. A noisy input signal is now filtered with a BPF to attenuate out-of-band noise.

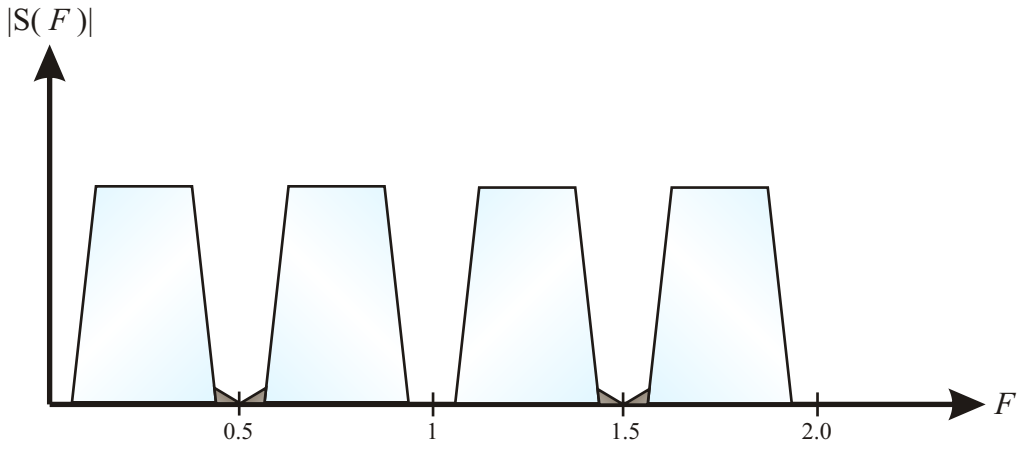


Figure 3.7: Minimal Aliasing Effects of a Sampled Noisy Signal. The band of interest is now minimally affected by the out-of-band noise.

band frequencies. One of the major drawbacks to using these types of filters is the high insertion loss and the delay introduced into the system.

Ideally sampled signals can be thought of as being multiplied by a train of impulses as described by the equation 3.2. One of the properties of the Fourier Transform is that multiplication in the time domain is analogous to a convolution in the frequency domain. This can be used to describe the sampled signal in

the frequency domain as shown in equation 3.4. The convolution of impulses can be rewritten as a summation of the frequency representation of the original signal shifted to intervals of the sampling frequency. These shifted replicas of the original signal are called spectral images and are the cause of aliasing as shown in figure 3.5 and allow for the down-conversion as seen in figure 3.7.

$$x_s(t) = x_c(t) \sum_{n=-\infty}^{\infty} \delta(t - \frac{n}{f_s}) \tag{3.2}$$

$$\begin{aligned} X_s(F) &= X_c(F) * f_s \sum_{k=-\infty}^{\infty} \delta(F - k) && (3.3) \\ &= f_s \sum_{k=-\infty}^{\infty} X_c(F - k) && (3.4) \end{aligned}$$

E_b/N_0 Requirement

Reducing noise sources, such as aliased noise, is important to meeting the E_b/N_0 requirement. The maximum value for this is specified as 10 dB Hz/sec. With this, one can calculate the noise spectral density, N_0, since the nominal power output is known to be -12 dBm and the given maximum data rate is 4 Mb/sec or 66 dB Hz (see Appendix A).

$$N_0 = \frac{f_{\text{data}} P_{\text{output}}}{E_b/N_0} = \frac{(66\text{dB Hz})(-12\text{dBm})}{10\text{dB Hz/sec}} = -88\text{dBm/Hz} \tag{3.5}$$

Knowing the maximum noise spectral density indicates the upper bound of the noise floor if the noise is stochastic, but it doesn't give much insight into the maximum power of spurious signals. Theoretically, a spur, or sharp peak in the spectral power measurement, is modeled as a single frequency with an infinite spectral density at that frequency and zero spectral density elsewhere. Practically speaking, this is never this case. Knowing the spectral density of a spur would

require knowing the width in the frequency spectrum which the spur spread. Measuring this for each spur is not only tedious but is dubious as discussed in section 7.1.1. For the purposes of this work, spurs are cautiously estimated to have 1 Hz width in the frequency spectrum; in reality it is often much greater. This would simply mean that in order to fulfill this requirement, no spur can be measured to exceed -88 dBm.

3.4 Digital Signal Processing Path

Much of the physical phenomena that characterize the space channel are best created through digital means due to their relative complexity. The high throughput that the sampling frequency of 56 MHz demands requires many fast, concurrent computations. This precludes the use of a single processor to perform the necessary operations. Instead, an Field Programmable Gate Array (FPGA) is used. This allows for the high-level of parallel processing required.

The data path proposed by JPL, shown in figure 3.8 and outlined in the JPL statement of work (Appendix A), splits the incoming signal into separate in-phase (I) and quadrature (Q) paths. The main advantage of doing this is that the main DSP path only needs to operate at half the sample frequency. The drawback to this is the data path is now twice as wide. Given that the current operating speeds of an FPGA easily meet the 56 MHz requirement, a simpler single data path approach was chosen as shown in figure 3.9.

It should be noted that the statement of work also requested the ability to control some of the operating parameters separately for the I and Q channels. It was later determined that such separation is done on an arbitrary basis as described by the statement of work allowing for no benefit in having separate control of the two channels. JPL relaxed this requirement and approved the single data path design.

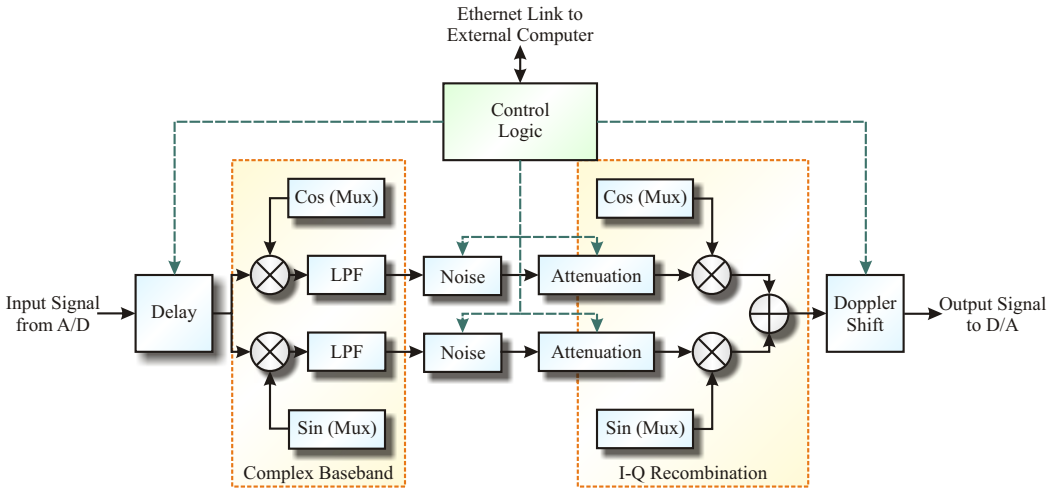


Figure 3.8: Block Diagram of the Proposed DSP Path.

The DSP path can be broken down into a few main areas as shown in figure 3.9. These areas correspond to the main space communication channel characteristics namely delay, the Doppler Effect, and AWGN. Each is discussed in further detail in the following subsections.

3.4.1 The Memory Controller

As mentioned previously, adding delay to the signal is essential to recreating the characteristics of the space channel. For this project, it was requested that the added delay range from less than 1 ms to greater than 2.5 seconds with a granularity of less than 1 ms. As shown in figure 3.9 the delay stage is the first to be implemented in the DSP path. The reason for this is to allow any changes in the parameters not related to delay to take immediate effect without having to wait the current delay setting.

In the original project description provided by JPL (Appendix A), it is requested that the delay be applied independently to both the I and Q channels. One possible benefit of having distinct delays for the I and Q channels might

Figure 3.9: Block Diagram of the Designed DSP Path.

be to provide a better emulation of multipath effects and fading. Aside from the inherent added complexity in creating two separate delay paths, the I and Q channels are discerned arbitrarily in the scheme presented–the I and Q channels might be quite different from test to test given the sample input signals making their separation almost problematic when designing a repeatable environment for which to test the signal. As was previously stated, it was decided to abandon the I and Q separation and implement a single delay path.

The implementation of the delay in the DSP path relies on a memory controller to store the incoming sampled signal for a discrete number of samples in a bulk memory bank. At the sample frequency of 56 MHz and an upper limit of 2.5 seconds for the delay, the total number of samples that needs to be stored is 140 million. With roughly two bytes per sample this comes to slightly greater than 256 megabytes of memory.

The memory controller is designed around the idea of a first-in-first-out (FIFO) type buffer. The sampled data coming in will fill a circular type buffer while the data fed out to the remainder of the DSP path will be pulled from an index that

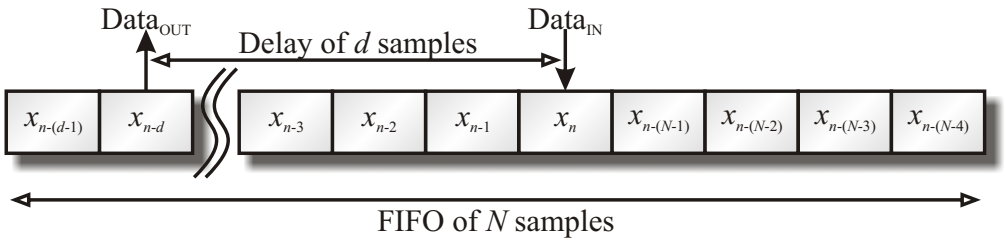


Figure 3.10: FIFO Implementation of a Delay Line [11]. The data written to the FIFO fills successive locations until it wraps around after N samples. Data read from the FIFO reads from successive locations with a distance from the write address equal to the delay desired.

follows the input index by the number of samples to be delayed.

The delay is to be changed dynamically during testing to accommodate the scenario of a continually changing distance over time between the source and the receiver. Given the discrete nature of the memory controller, any change in delay will result in a loss or repeat of data causing a discontinuous signal to be output. To alleviate this issue the memory controller has been designed with the ability to clock data out at a different rate than it is being clocked in changing the delay time while keeping the signal continuous. The memory controller was designed to handle this two different ways: allow for different clocks at the input and output, and allow for multiple samples to be extracted at any given time. While the first approach is much more straight forward, it does pose problems with generating clock signals. This will be discussed further in the clock management section of this chapter. The second approach gives multiple samples to be used in the following interpolation stage and will be further discussed in section 3.4.2.

3.4.2 Interpolation Finite Impulse Response (FIR) Filter

As discussed in the previous chapter, the Doppler Effect can be thought of in the frequency domain as having a scaling factor, namely $1 \pm \frac{v}{c}$ where v is relative velocity between the source and the receiver and c is the speed of light. Since

this scaling factor amounts to a shift in frequency proportional to the original frequency itself, a simple static shift in frequency will not suffice to accurately reproduce the Doppler Effect. Also, the Doppler Effect occurs from the fact that there is a changing distance, and thus a changing delay, between the source and the receiver.

$$\mathscr{F}^{-1}\left\{\frac{1}{|\alpha|}X\left(\frac{f}{\alpha}\right)\right\} = x(\alpha t) \tag{3.6}$$

Viewed in the time domain instead of the frequency domain, as in equation 3.6, one can see that a scaling in the frequency domain translates to a corresponding scaling in the time domain. Essentially what this means is that the Doppler Effect can be reproduced by simply pulling data from the delay buffer slightly faster or slightly slower than it is being put into the buffer. This "throttling" of the data achieves both the desired frequency scaling as well as the desired dynamically changing delay.

There is some hardware difficulty when changing the frequency of the output sample rate dynamically internal to an FPGA. The main difficulty arises from the fact that most clock signals entering an FPGA are run through a Digital Clock Manager (DCM) to correct the duty cycle and condition the signal to be used on the global clock network. The DCMs require a reset with any input frequency change in order to establish a lock with their internal phase lock loops (PLLs). This reset creates a discontinuity in the output signal–a result that the throttling technique was intended to eliminate in the first place.

To change the output data rate without changing the sample clock frequency requires a dynamic interpolation filter. Figure 3.11 shows an example of this approach. Ideally, the value between samples is determined successively getting further away from one sample point with each clock. This gradual change in interpolation position effectively changes the output data rate of the sampled input signal. This method also enables delays that are a fraction of a sample

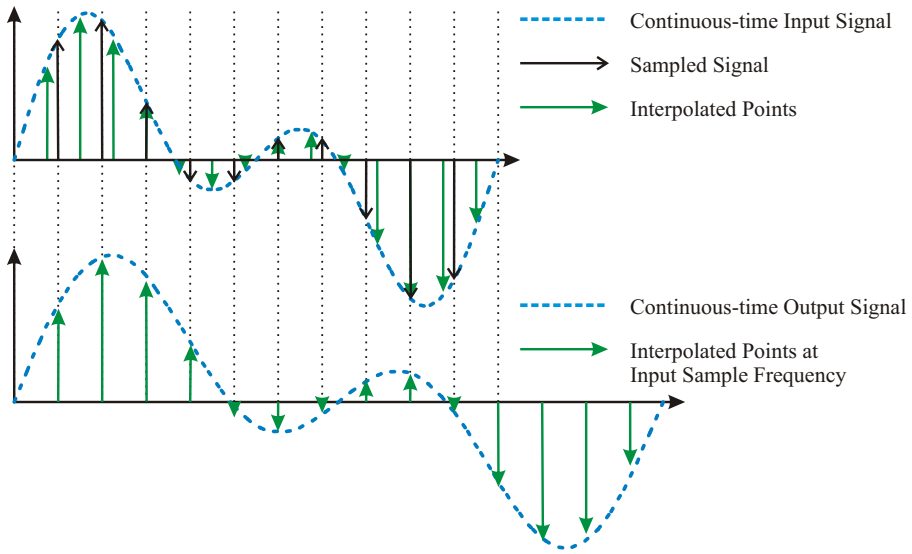


Figure 3.11: Using Interpolated Points to Time-Scale a Signal. The sampled input signal is used to find points between samples that will construct a time-scaled version of the original signal that can be output without changing the sample frequency.

rather than being forced to a discrete sample delay as solely using a FIFO does.

There are multiple approaches to the design of an interpolation filter. The most direct approach is to determine a mathematical representation of an arbitrary point in time given the sampled data. This is shown in equation 3.9. Passing the continuous-time representation of the sampled signal through a lowpass filter (LPF) with a cutoff frequency at $f_s/2$, which assumes the input signal doesn't contain any frequencies above this that might be aliased, yields a reconstructed continuous-time signal. Thus, any point in this reconstructed signal can be determined based on the sampled input signal given filter coefficients that correspond to the time of interest desired. This approach is called Sinc interpolation.

$$x_c(t) = x_s(t) * LPF \tag{3.7}$$

$$= \sum_{n=-\infty}^{\infty} x_s[n] \frac{\sin[(t - \frac{n}{f_s})f_s]}{(t - \frac{n}{f_s})f_s} \tag{3.8}$$

$$= \sum_{n=-\infty}^{\infty} x_s[n] \text{sinc}[(t - \frac{n}{f_s})f_s] \tag{3.9}$$

The major concern with the Sinc interpolation method is, as one can see in equation 3.9 is that to get the exact result of the interpolated points, an infinite number of data samples is required. There are other interpolation methods that trade off accuracy for simplicity such as the zero-order hold, linear interpolation, and raised cosine interpolation. Each of these are more cost effective, with respect to the computational effort required, in producing an estimation, but none can approach the accuracy of the Sinc method.

Since the Sinc interpolation coefficients gradually approach zero as they extend to infinity, a finite length filter can be used to approximate the sinc interpolation in order to produce the desired precision. This allows for the use of a finite impulse response (FIR) filter implementation. Although the length can be adjusted to meet any precision desired, the causal nature of an actual FIR implementation results in a delay, measured in samples, of half the filter size. Thus, there could be the possibility of conflicting constraints in acceptable delay and acceptable accuracy.

The resolution of the interpolation filter–the number of points that the filter can distinguish between samples–can be determined by the desired resolution of the Doppler Effect: 1 Hz. This in itself is not enough to calculate an acceptable resolution since it has already been determined that the Doppler shift is dependent upon the input signal frequency. The scaling factor $\alpha = 1 \pm \frac{v}{c}$ can be determined for the center frequency, f_c, of 70 MHz when Δf is equal to 1 Hz

using this simple formula:

$$\Delta f = (1 - \alpha) f, \qquad f = f_c \tag{3.10}$$

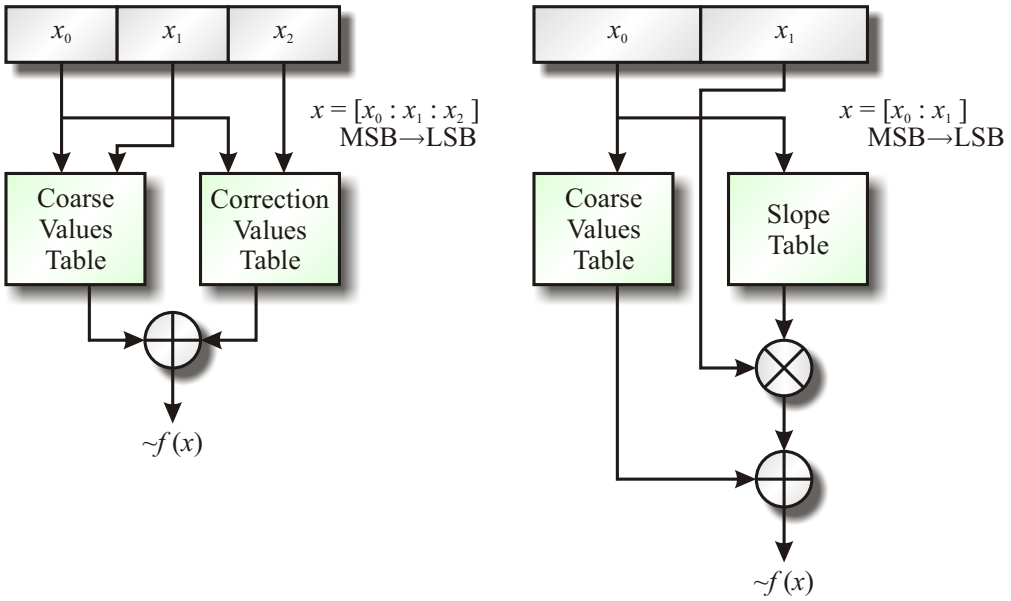
The number of interpolation points, N, must be $|1 - \alpha|^{-1}$ to achieve proper time-scaling. Thus, N is simply 70 million.

Storing that many different values for each of the filter's coefficients is somewhat impractical with the capabilities of today's FPGAs. Often, a bipartite table system is used to reduce the size if data stored for the coefficients into something more manageable while assuring a certain degree of accuracy [18]. Bipartite tables basically stores two coefficients: one for a very coarse approximation and one for a correction factor. The sum of the two table parameters will give a very close approximation while only requiring a simple adder and much less table space. The maximum table compression is found with this approach when there are $N^{2/3}$ entries in each table. This comes to approximately 170 thousand entries in each table. While there are FPGAs with the capability of storing 340k values for each coefficient, adding a little more complexity can reduce the table size significantly.

Most FPGAs have hardware multipliers built in that can be used to create linear approximations by storing a table of values of the desired function along with another table for the slope. The value for any point can then be approximated by the sum of the stored value and the value linearly interpolated given the slope and the distance from the stored value. Figure 3.12 shows the bipartite and linear interpolation methods.

3.4.3 Digital Single Side Band Modulation

In the previous sections, it has been shown that the Doppler Effect shifts a given input frequency by an amount dependent on that frequency itself and a



Bipartite table method.

Linear interpolation method.

Figure 3.12: Using Tables to Approximate a Function. The bipartite method of approximating a function is considerably simpler to implement but generally requires much larger tables than does the linear interpolation approach [18].

scaling factor. Using a dynamic interpolation filter, this scaling factor can be applied to a sampled signal. So far, the inherent down-conversion of the sampled signal in the sampling process has been not been addressed. This down-conversion results ins all frequencies being shifted down by from the original center IF of 70 MHz to the 14 MHz IF. Thus, the sampled signal can be written generally as:

$$S_{in}(f) = S_{IF}[f - (f_c - f_{IF})], \quad f \geq 0 \tag{3.11}$$

$$S_{in}(f) = S_{IF}[f + (f_c - f_{IF})], \quad f < 0 \tag{3.12}$$

where S_{IF} is the sampled signal at an IF of 14 MHz, f_c is the actual center frequency of transmission, and f_{IF} is 14 MHz. When this is scaled by the Doppler Effect with the scaling factor $\alpha = 1 \pm \frac{v}{c}$, the result is:

$$S_{in}(\alpha f) = S_{IF}[\alpha f - sgn(f)\alpha(f_c - f_{IF})] \tag{3.13}$$

Thus the change in frequency (δf) is,

$$\Delta f = (1 - \alpha)f - sgn(f)(1 - \alpha)(f_c - f_{IF}) \tag{3.14}$$

in the digital domain. Given α, the expression $(1 - \alpha)(f_c - f_{IF})$ is a constant frequency shift not dependent on the input frequency, f. This shift is applied to all positive frequencies. A shift of all positive frequencies in one direction and all negative frequencies the other is called single side-band modulation (SSB).

Single side-band modulation in the digital domain can prove slightly more difficult than in the analog domain when the desired shift is less than the width of the band of interest and the band of interest consumes more than half of the total bandwidth of $f_s/2$ where f_s is the sample frequency. Much of the difficulty occurs from aliasing effects when frequencies move above $f_s/2$ or below DC. It is common to multiply the signal by a cosine function to shift the frequency both up and down, then filter off the undesired direction. When shifting by an

amount that leaves the two overlapping, it becomes difficult to discern the two thus making filtering impossible.

One popular method for achieving SSB modulation in is the Hartley Method [16]. The basic diagram for the Hartley Method is shown in figure 3.13. This method employs a Hilbert Transform to first shift the phase of the signal by $-90°sgn(f)$ for all frequencies. Thus, signals A and B can be described by the following equations:

$$\begin{aligned} A(F) &= X(F) && (3.15) \\ B(F) &= e^{-j\frac{\pi}{2}sgn(F)}X(F) \\ &= -jsgn(F)X(F) && (3.16) \end{aligned}$$

C and D can then be described by:

$$\begin{aligned} C(F) &= \frac{1}{2}X(F)(e^{j2\pi F_c n} + e^{-j2\pi F_c n}) \\ &= \frac{1}{2}\left[X(F-F_c) + X(F+F_c)\right] && (3.17) \\ D(F) &= \frac{1}{2}sgn(F)X(F)(e^{j2\pi F_c n} - e^{-j2\pi F_c n}) \\ &= \frac{1}{2}sgn(F)\left[X(F-F_c) - X(F+F_c)\right] && (3.18) \end{aligned}$$

Thus $Y(F) = C(F) + D(F)$,

$$\begin{aligned} Y(F) &= \frac{1}{2}[1+sgn(F)]X(F-F_c) + \frac{1}{2}[1-sgn(F)]X(F+F_c) \\ &= \begin{cases} X(F-F_c), & F \geq 0 \\ X(F+F_c), & F < 0 \end{cases} && (3.19) \end{aligned}$$

The Hartley Method had one major drawback: any discrepancies in the magnitude responses of the Hilbert Transform and the matched delay will result in allowing an undesired shift direction which does not perfectly cancel. Implement-

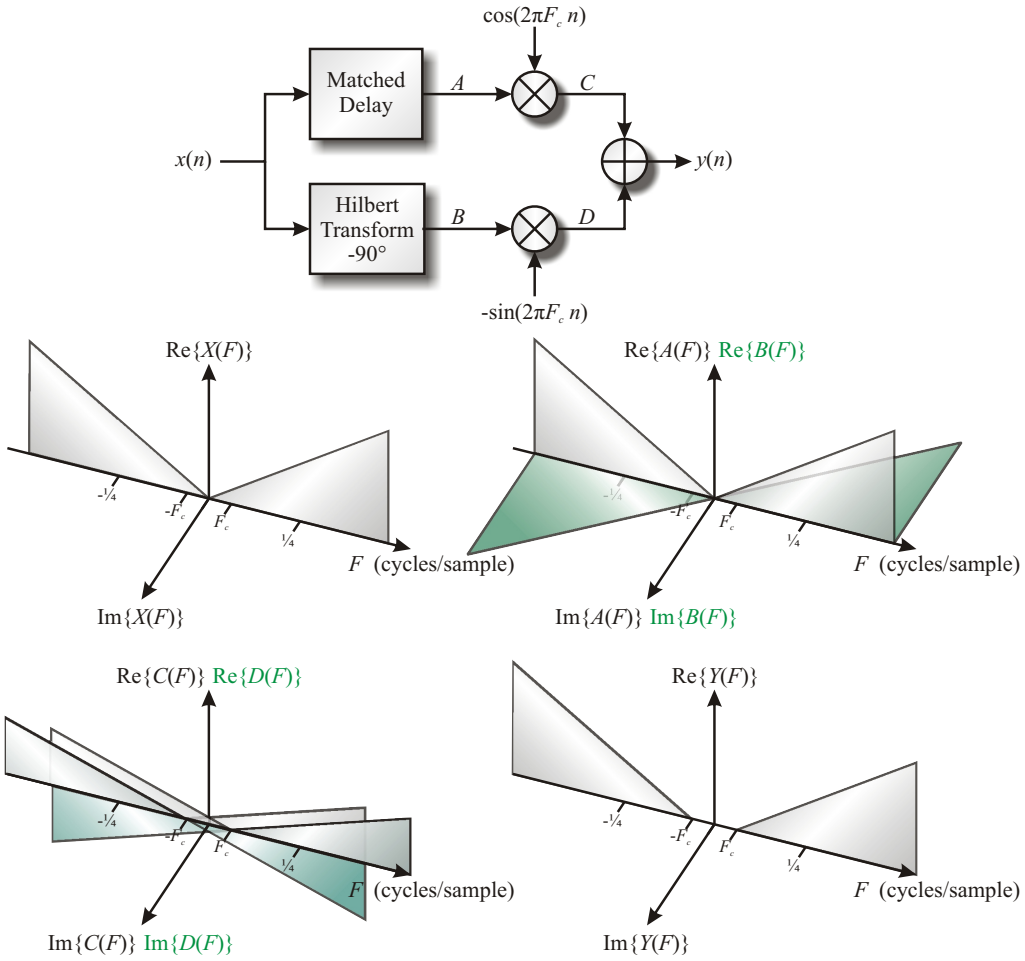


Figure 3.13: Hartley Method for SSB Modulation [16]. This method requires a Hilbert Transform whose response perfectly cancels the all pass match delay.

ing this transform with an FIR filter, as it most likely would be, will result in an imperfect response due to the finite nature of the filter. Thus, some amount of error is to be expected. However, the relative simplicity of this method makes it attractive for implementation.

Another popular approach to SSB modulation is the Weaver Method [16]. This method is detailed in figure 3.14. From the diagram, points A and B are passed through a lowpass filter with a cutoff frequency of $\frac{1}{4}$ cycles/sample. As

such, these points can be expressed as follows:

$$A(F) = \begin{cases} X(F-\frac{1}{4}) + X(F+\frac{1}{4}), & |F| < \frac{1}{4} \\ 0, & \text{otherwise} \end{cases} \tag{3.20}$$

$$B(F) = \begin{cases} -j\left[X(F-\frac{1}{4}) - X(F+\frac{1}{4})\right], & |F| < \frac{1}{4} \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

Points C and D then follow to be:

$$\begin{aligned} C(F) &= \frac{1}{2}\left[A(F-F_c-\frac{1}{4}] + A(F+F_c+14)\right] \\ &= \begin{cases} \frac{1}{2}\left[X(F-F_c-\frac{1}{2}) + X(F-F_c)\right], & F_c < F < F_c + \frac{1}{2} \\ \frac{1}{2}\left[X(F+F_c) + X(F+F_c+\frac{1}{2})\right], & -F_c - \frac{1}{2} < F < -F_c \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.22}$$

$$\begin{aligned} D(F) &= -j\frac{1}{2}\left[B(F-F_c-\frac{1}{4}] - B(F+F_c+14)\right] \\ &= \begin{cases} \frac{1}{2}\left[-X(F-F_c-\frac{1}{2}) + X(F-F_c)\right], & F_c < F < F_c + \frac{1}{2} \\ \frac{1}{2}\left[X(F+F_c) - X(F+F_c+\frac{1}{2})\right], & -F_c - \frac{1}{2} < F < -F_c \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.23}$$

Thus, summing the two yields the result, $Y(F) = C(F) + D(F)$:

$$Y(F) = \begin{cases} X(F-F_c), & F_c < F < F_c + \frac{1}{2} \\ X(F+F_c), & -F_c - \frac{1}{2} < F < -F_c \\ 0, & \text{otherwise} \end{cases} \tag{3.24}$$

With the periodic spectral images at every integer value in the digital frequency spectrum, this result matches exactly to that of the Hartley Method as one would expect.

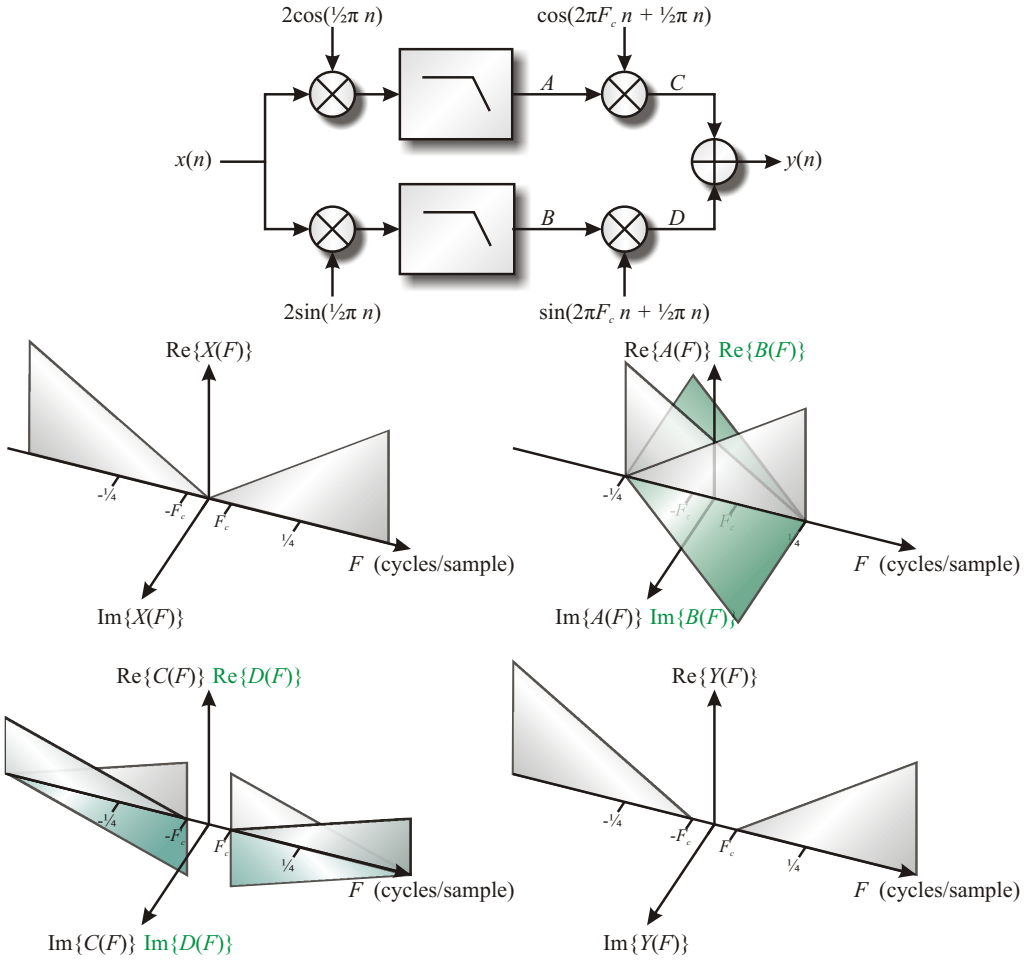


Figure 3.14: Weaver Method for SSB Modulation [16]. This method requires two ideal LPFs with corner frequencies at $F = \frac{1}{4}$.

The major drawback of the Weaver Method is similar to that of the Hilbert Method; the required LPF cannot be perfectly implemented. Some transition band will be present allowing a slight attenuation of the higher frequencies, but more notably, will cause aliasing for the imperfections in the stop band.

A final approach to consider for SSB modulation is the Balanced Hartley Method, otherwise known as the Modified Hartley Method [17]. This approach is shown in figure 3.15. Here, fractional Hilbert Transforms [21] are used to ensure identical responses for each path. To start the analysis, points A and B can be described as the following after being passed through the Fractional Hilbert

Transforms.

$$A(F) = e^{j\frac{\pi}{4}sgn(F)}X(F) \tag{3.25}$$

$$B(F) = e^{-j\frac{\pi}{4}sgn(F)}X(F) \tag{3.26}$$

Points C and D are then described below after being multiplied by the sine and cosine functions similar to the normal Hartley Method but with a correction in the phase to accommodate the fractional Hilbert Transforms.

$$\begin{aligned} C(F) &= \frac{1}{2}e^{j\frac{\pi}{4}[sgn(F)-1]}X(F-F_c) + \frac{1}{2}e^{j\frac{\pi}{4}[sgn(F)+1]}X(F+F_c) \\ &= \begin{cases} \frac{1}{2}X(F-F_c) + j\frac{1}{2}X(F+F_c), & F \geq 0 \\ -j\frac{1}{2}X(F-F_c) + \frac{1}{2}X(F+F_c), & F < 0 \end{cases} \end{aligned} \tag{3.27}$$

$$\begin{aligned} D(F) &= j\frac{1}{2}e^{-j\frac{\pi}{4}[sgn(F)+1]}X(F-F_c) - j\frac{1}{2}e^{-j\frac{\pi}{4}[sgn(F)-1]}X(F+F_c) \\ &= \begin{cases} \frac{1}{2}X(F-F_c) - j\frac{1}{2}X(F+F_c), & F \geq 0 \\ j\frac{1}{2}X(F-F_c) + \frac{1}{2}X(F+F_c), & F < 0 \end{cases} \end{aligned} \tag{3.28}$$

The final result is:

$$Y(F) = \begin{cases} X(F-F_c), & F \geq 0 \\ X(F+F_c), & F < 0 \end{cases} \tag{3.29}$$

As mentioned before, this approach gets around the major drawback of the standard Hartley Method by ensuring both paths have the same, even if imperfect, responses for the Hilbert Transform. This approach also isn't burdened with the drawback of the Weaver Method since this method does not depend on LPFs to eliminate unwanted aliasing. As with all three methods, this will be subject to any aliasing inherent in the shifting of frequencies.

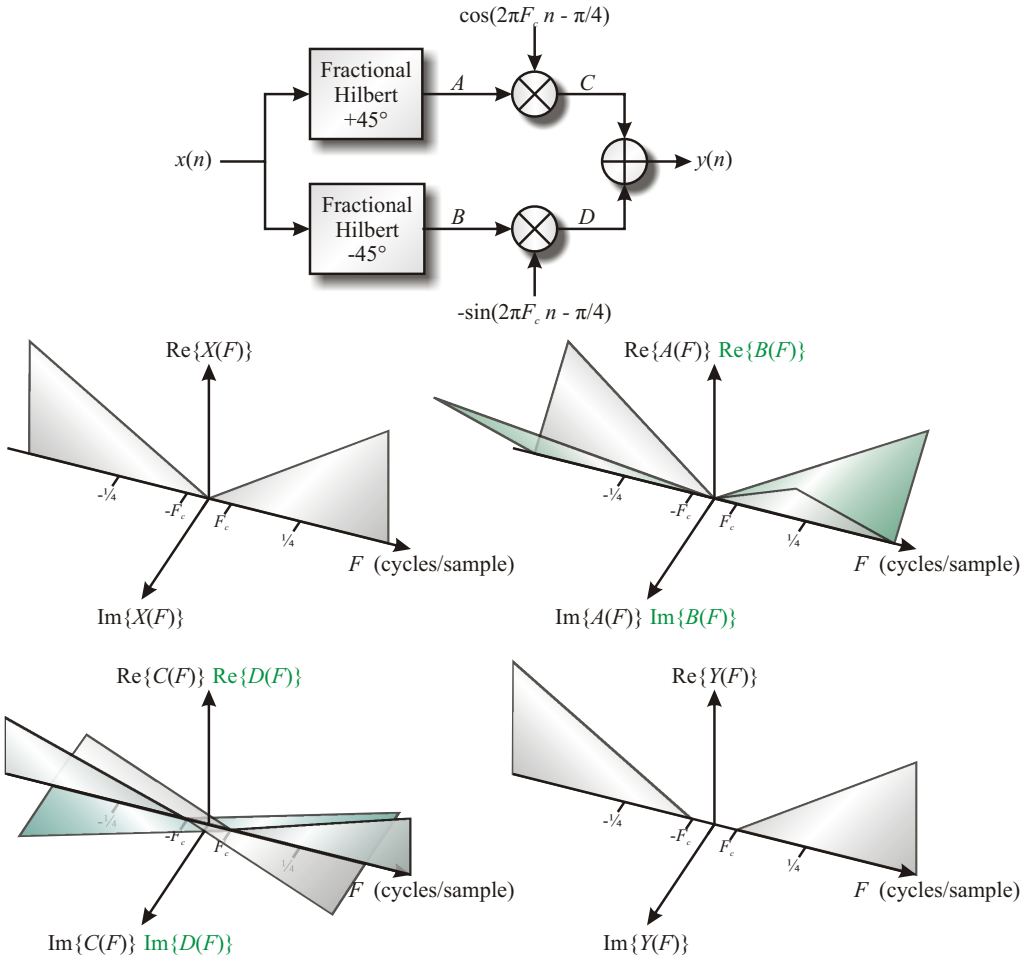


Figure 3.15: Balanced Hartley Method for SSB Modulation [17]. This method ensures an identical response on each path by passing both through Fractional Hilbert Transforms.

3.4.4 Digital Attenuation

When dealing with a space communication channel, the medium is often considered lossless for electromagnetic transmissions. This is based on the assumption that the medium is a perfect vacuum. In reality, space is not a perfect vacuum, being populated by sparse particles. Therefore, the assumption of a lossless medium is not the best model in all space communication scenarios [22]. Also, the power of a signal is spread over an ever increasing area as the signal propagates causing the power that reaches the receiver to be proportional to the

inverse square of the distance. To recreate the conditions of an imperfect medium and the power loss due to distance, attenuation is digitally applied to the signal in the DSP path.

Attenuation is achieved by simply multiplying the signal data by a constant within the range [0, $2^N - 1$]. Here, N is the number of bits in the digital multiplicand. A value of 1.0 is assigned 2^{N-1} which allows for digital amplification up to approximately 2.0. While emulating a dynamic scenario, the amount of attenuation applied to the signal may fluctuate over time. If the dynamic attenuation changes are set correctly, the fluctuations realistically mimic multipath interference on the signal. As a result, the ability to vary the attenuation with time allows accurate emulation of the effects of Rician fading [12].

3.4.5 White Gaussian Noise Generator

In space communications, the signal receives the addition of noise from two main sources: natural sources such as nearby celestial sources, and noise inherent in the electronic systems conducting the communication. The latter is most often the stronger source coming mainly from the input gain that attempts to automatically keep the input signal at a constant power known as the automatic gain control (AGC). In order to accurately reproduce such noise sources, a Gaussian random process is used for the noise model. White noise is approximated, for the sake of implementation, using a pseudo random noise generator with a selectable mean and variance. A true Gaussian process results in values that extend from negative infinity to positive infinity which, of course, cannot be realized in discrete digital values. Therefore, the digital Gaussian process is truncated to a set number of standard deviations, a value of 4σ for this application, to eliminate the extreme positive and negative noise peaks.

To implement a digital noise generator in the RF Channel Emulator, an existing Adaptive White Gaussian Generator noise core developed by Xilinx is used.

This core produces a pseudo-random number with a zero-mean and 4.8 standard deviations. This pseudo-random number is then scaled to the desired mean and variance. This adjusted noise value is added to the signal in the digital signal path.

3.4.6 Digital Output Filter

The space communication medium is near perfect–passing all frequencies equally well. It is important, when emulating the space channel, to have this near perfect response as well. As the signal passes through the digital path, it maintains, in theory, a fairly flat frequency response. In reality, there are some minute discrepancies. When dealing with the analog components, this imperfect reality is distinctly more pronounced. The digital-to-analog converter (D/A) will also affect the output magnitude due to the sinc response inherent in a zero-order hold topology. The output filter is necessary to alleviate any deviations from a flat response for the entire band of interest regardless of their origin.

The output filter, which is attached to the end of the digital path, is best realized using an FIR filter like many of the components in the path. Using an odd number of symmetric coefficients, a linear phase response can be achieved to assure that phase as well as magnitude is minimally affected by the emulator. The coefficients are determined by measuring the response of the entire system, $H_{system}(F)$ and setting the response of the filter to $|H_{filter}(F)| = 1/|H_{system}(F)|$.

3.5 Digital and Analog Output Stage

The output stage reconstructs the analog equivalent of the digital signal. This was originally proposed to be realized as is shown in figure 3.16 [22]. The FPGA feeds the D/A with the digital signal data. The D/A upsamples the data stream and passes this through an internal, digital lowpass filter. Using a mixer, this

signal is then up-converted from an IF of 14 MHz to that of 70 MHz.

Figure 3.16 also shows the harmonic noise inherent in any system with quantized values. As one can see by the possible locations of the harmonics, as shown in the figure, these mainly fall within the band of interest. This is due to the fact that the harmonics are digital constructs and fold per the rules of spectral images along the digital frequencies of $n/2$. The upsampling performed by the D/A does leave some space for the harmonics to fall that is out-of-band, but can be improved upon. The harmonics themselves cannot be eliminated but only reduced by increasing the data width.

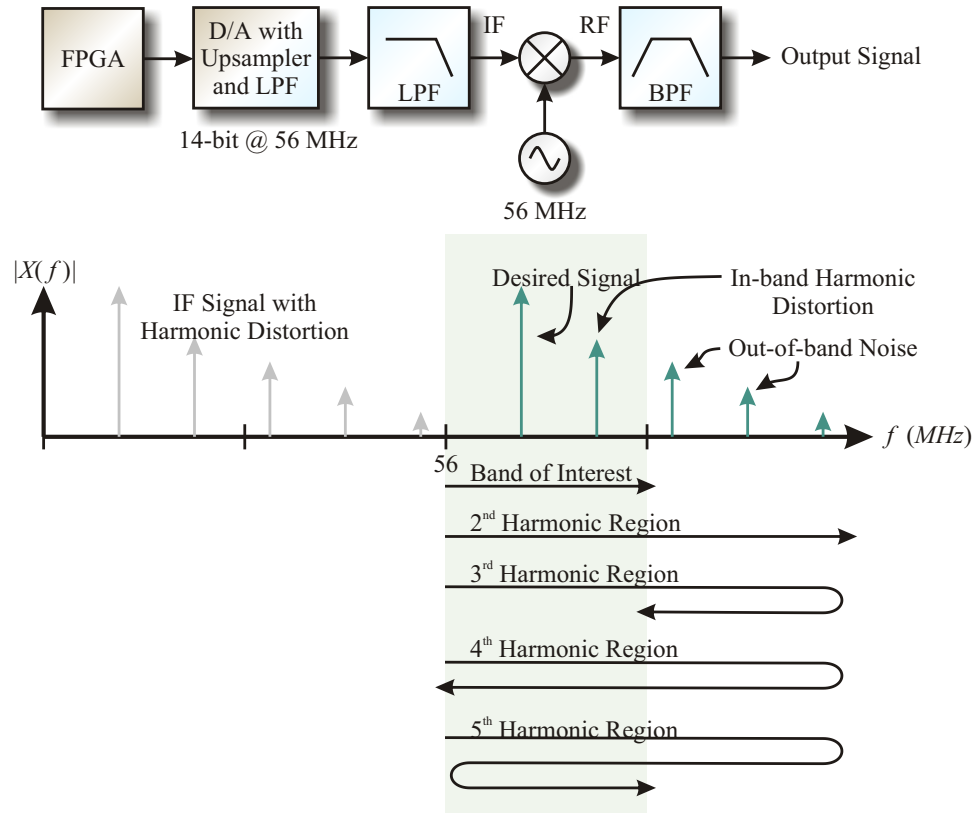


Figure 3.16: Block Diagram of Original Analog Output Stage with Quantization Harmonic Noise. A quantized output digital signal can introduce harmonics which fall both inside and outside the band of interest.

Figure 3.17 shows a modified version of the output stage that includes a digital upsampler in the FPGA which essentially creates more out-of-band space for the harmonic noise to reside. This reduces the total harmonic noise inherent in quantization. This topology also minimizes the number of analog components needed in the system thus greatly reducing any analog noise introduced.

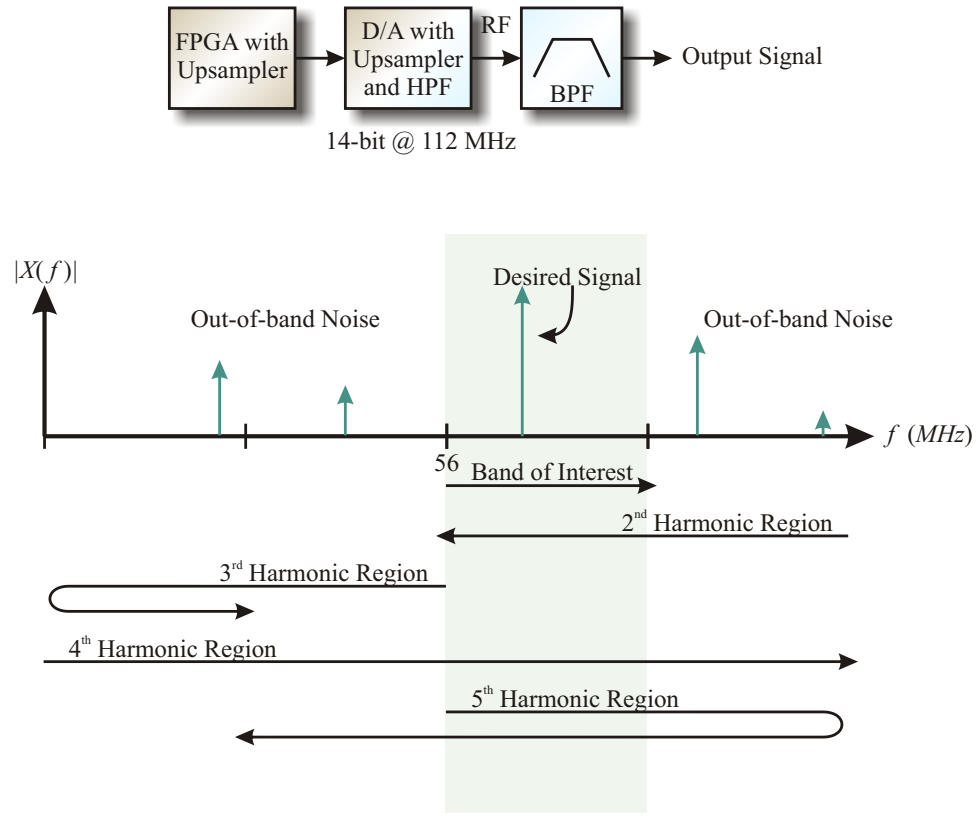


Figure 3.17: Block Diagram of an Improved Analog Output Stage with Reduced Quantization Harmonic Noise. More out-of-band space is given for harmonic noise to fall using an upsampler in the FPGA.

The design of the upsampler is fairly straight forward. Initially a digital signal, $x(n)$, has zeros interspersed between each sample. This is the equivalent of:

$$y(n) = x(n/2)\frac{1}{2}[1 + cos(\pi n)] \tag{3.30}$$

which yields a frequency response of:

$$\begin{aligned} Y(F) &= \tfrac{1}{2}\left[X(2F) + \tfrac{1}{2}X[2(F-\tfrac{1}{2})] + \tfrac{1}{2}X[2(F+\tfrac{1}{2})]\right] \\ &= X(2F) \end{aligned} \tag{3.31}$$

This is illustrated in figure 3.18.

The D/A is designed with a selectable highpass or lowpass filter after the upsampling to allow one or the other to be used as the output. In the case of the original output stage design, the lowpass filter was needed to keep the signal at the 14 MHz IF. However, with an upsampler and highpass filter in the FPGA, the data reaching the D/A has an IF shifted by 28 MHz to create a 42 MHz IF. Selecting the lowpass filter on the D/A, in this case, would keep the 42 MHz IF while also keeping the mirror image of the signal in the frequency spectrum. Selecting the highpass filter allows for the signal to be shifted again by 28 MHz effectively up-converting it to the 70 MHz center frequency that is desired.

The output of a D/A is commonly a zero-order hold attenuating spectral images according to a $sin(x)/x$ response. This is illustrated in figure 3.19. The spectral images, if not attenuated by a bandpass filter, might cause problems at the input of a transceiver [4]. A common issue is out-of-band noise and spectral images saturating the input of the transceiver.

3.6 Clock Management

The clock distribution network serves many purposes in the RF Channel Emulator. It contains the embedded processor system clock to execute the code which handles dynamic system parameter changes. This same clock is also responsible for the communication with and control of the external memory module used to implement the delay. Code execution is necessary to initialize the Dynamic Digital Synthesizer (DDS) which is used to generate the sampling clocks for both

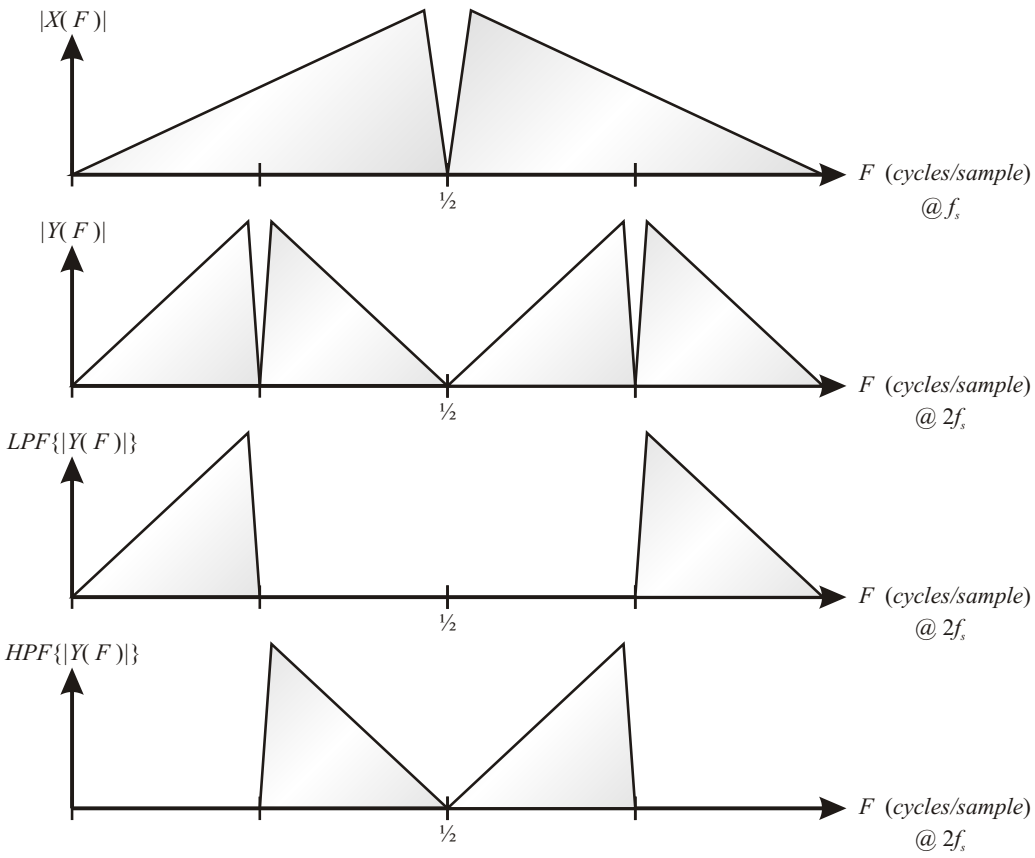


Figure 3.18: Upsampling with Highpass and Lowpass Filters.

the A/D and D/A components.

The DDS is essentially a sinusoidal wave generator which utilizes a lookup table and built in D/A converter along with a phase incrementor which allows a wide range of frequencies to be generated while also ensuring a fine granularity in frequencies. The DDS was chosen given the requirement of a non-standard frequency–a frequency for which off-the-shelf components are not easily obtainable–to be used for sampling and recreating the input and output analog signals respectively.

Use of the DDS can inadvertently introduce noise in various forms into the sampled and reconstructed signals. Some of this noise is due to deterministic

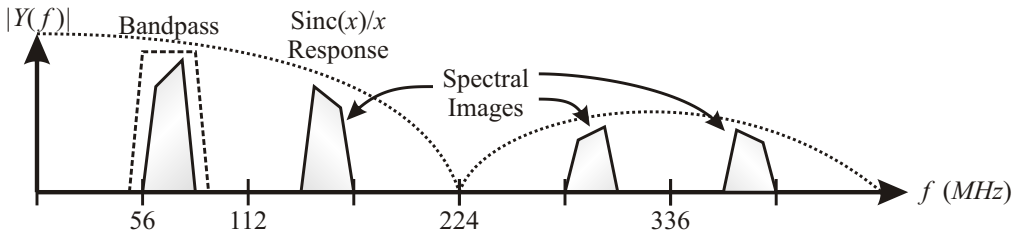


Figure 3.19: D/A Spectral Output with Required Bandpass Response. The bandpass filter attenuates any spectral images created by the D/A.

jitter which has multiple sources: the quantization error introduced from the 14-bit D/A in the DDS, any PLLs used for clock multiplication within the DDS itself, and, to a much lesser degree, induced noise from nearby high speed digital switching circuits and traces. The design of this portion of the system must be done with care to eliminate as much deterministic noise as possible. (See section 7.1.3 for more discussion pertaining to the effect of deterministic jitter in sampling.)

The second form of noise that can be introduced into the sampling and reconstruction clocks is stochastic jitter, commonly quantified as random phase noise. This, too, can be a by product of any PLL used in the DDS itself. More importantly, this is best reduced at its initial source: the clock driving the DDS. It is imperative that this clock source be as "clean" as possible. Maximum jitter specifications can be found in figure 3.20 [14].

A simplistic model for the effect of random phase noise on a sampled signal can be understood by a quick look at the sampling theorem [6]. Here, the sampled signal is expressed as the clock signal, or an infinite series of impulses in time, multiplied with the continuous-time signal.

$$x_s(t) = x(t) \sum_{k=-\infty}^{\infty} \delta(t-k) \tag{3.32}$$

This multiplication in the time domain can be be viewed as a convolution in the

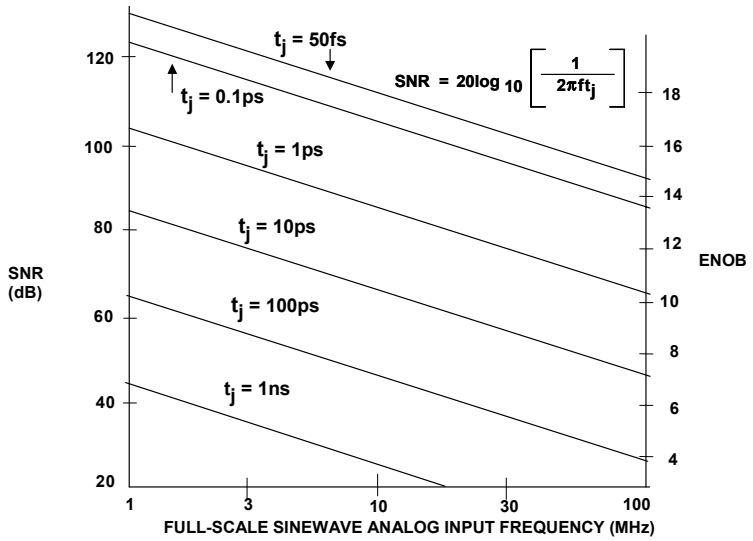


Figure 3.20: Theoretical SNR and effictive number of bits (ENOB) Due to Jitter vs. Input Frequency[14].

frequency domain of the clock spectrum with the signal spectrum. Figure 3.21 shows this representation. As can be seen, the phase noise in the clock is incorporated into the sampled signal which reduces the spectral resolution. Again, this demonstrates the need for a "clean" clock source. It should be noted that this simplistic model really only holds in practice for random clock jitter. Some experimental evidence of this can be found in section 5.1.4.

The final aspect of the clock distribution network that cannot be overlooked is the necessity to maintain synchronization between the various clocked components. This includes synchronization between the external components such as the A/D and D/A as well as between internal components operating from different clock sources. These clock sources could be potentially out of phase with each other or running at different frequencies altogether such as the main processor clock and the sampling clocks. This requirement can be fulfilled in the design through proper identification of in-phase clock sources and through internal FPGA logic tolerant of differences in phase and frequency.

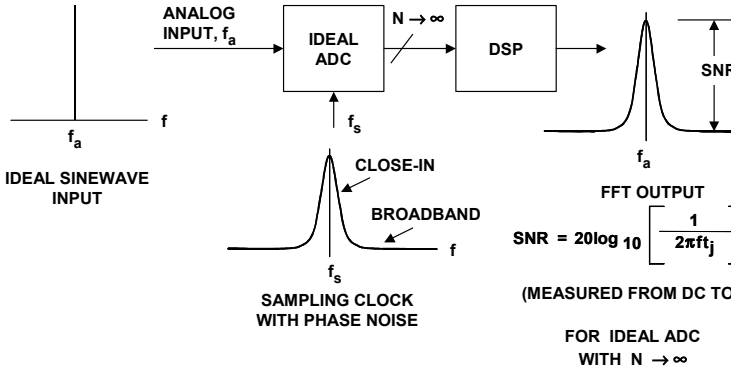


Figure 3.21: Effect of Clock Phase Noise on Ideal Digitized Sinewave[14].

Chapter 4

System Comparisons

For many communication applications, simulators are more than adequate to handle the needs of characterizing the performance of a given communication protocol. With today's computer systems, robust software can be written to produce the effects of noise, delay, and Doppler among others. However, software simulations are inadequate when testing the performance of real-time systems implemented in hardware. Despite the current speed of processors, the processing required to handle the DSP computations involved in reproducing the desired effects in real-time exceeds their ability. These processors, although very good at serial computing, cannot handle the extreme parallel processing involved in this application. Thus, hardware designed for this specific task is required. Such systems, as one might expect, are rare. One such system found with similar capabilities as the RF Channel Emulator is the SLE700 Satellite Link Emulator produced by dBm Corp.

The SLE700 is a satellite link emulator that produces the effects of a channel common in satellite communications [2]. The channel emulation is achieved by producing propagation delay, attenuation, flat fading, phase shift, and Doppler shift. The emulator allows for bidirectional or earth-to-satellite transmission paths.

The SLE700 is designed to be controlled by the front panel or through a

Figure 4.1: Front Panel of the SLE700 Satellite Link Emulator.

computer connected through an Ethernet LAN. The emulator operates in either static mode or in dynamic mode. The static mode allows the user to set a fixed set of conditions such as delay or Doppler shift. In dynamic mode, the hardware is equipped with preprogrammed satellite trajectories including low earth orbit, medium earth orbit, geostationary, and geosynchronous orbits. A simplified block diagram of the signal path of the SLE700 emulator is shown in figure 4.2.

The operation of the SLE700 is fairly straight forward. First, this input signal is down converted to using a direct digital synthesizer (DDS) and a mixer to baseband. Next, the baseband signal is passed through a lowpass filter to reduce aliasing. After the signal is buffered, it is digitized using a 12-bit analog-to-digital converter. The digital signal is then passed through a large first-in-first-out (FIFO) buffer to implement the delay. The data from the FIFO is reconstructed as an analog signal using a digital-to-analog converter. The analog output is then buffered and passed through a mixer to be up converted using another DDS back to the desired output center frequency. This is where a slight shift in frequency can be introduced. Finally, the signal is buffered and passed though a variable attenuator which sets the final output to the desired power level to emulate distance and fading.

The SLE700 comes with proprietary, GUI-based software that allows a user to program orbit and ground station coordinates, path loss models, and orbit models [22]. The user input is used to generate data files which are sent to the

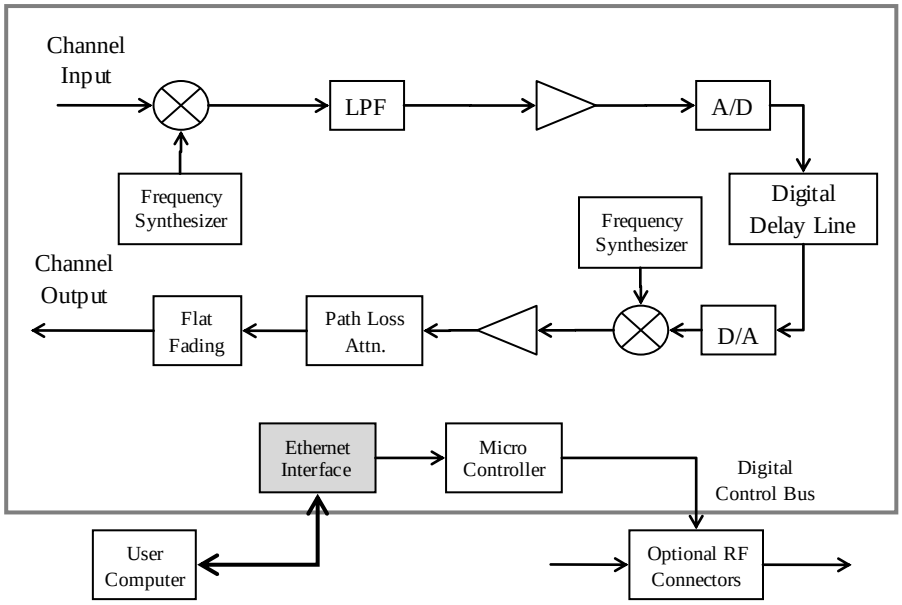


Figure 4.2: dBm SLE700 Satellite Link Emulator Block Diagram [2].

SLE700 emulator where they are executed to produce the desired communication path.

Despite the fact that the SLE700 and the RF Channel Emulator are similar systems, some notable differences are present. Many differences are in the way certain channel characteristics are emulated. While all the channel characteristics are implemented in the RF channel emulator in the digital path, it is only the delay that the SLE700 uses the the digital path for. The order in which the major channel effects are implemented remains the same.

The reliance of a simplified data path in the SLE700 force it to use analog components for many of the channel effects. This includes the introduction of noise. While the RF Channel Emulator has a built in Gaussian white noise generator and a dedicated input channel for an external noise generator, the SLE700 is only capable of introducing noise through an external noise generator. This means no noise is available without a separate piece of hardware to generate it.

Specifications	RF Channel Emulator	SLE700
Center Frequency	70 MHz	70 MHz
3 dB RF Bandwidth	24 MHz	45 MHz
Number of Channels	1	1, 2, or 4
Input Power	-7 dBm max.	-10 dBm max.
Output Power	-12 dBm max.	-10 dBm max.
Delay	10 μs to 2 s	100 μs to 700 ms
Delay Resolution	1 ps	1 ns
Doppler Shift	$\pm$3.5 KHz	$\pm$1.0 MHz
Doppler Resolution	0.1 Hz	0.1 Hz
True Doppler Shift	Yes	No
Attenuation	0 dB to 50 dB	0 dB to 40 dB
Noise	AWGN	None
Control	Software GUI	Front Panel, Software GUI
Interface	Ethernet	Ethernet

Table 4.1: Specification Comparison between the RF Channel Emulator and the SLE700.

The construction of the two emulators is not the only difference; the intended use of the two systems are slightly different as well. While the SLE700 emulator is intended for communication between earth and orbiting satellites, the RF channel emulator is intended for communication between the earth and spacecraft as far as the moon or between two spacecrafts. Therefore, the specifications of each system differ slightly. These differences are summarized in table 4.1.

Chapter 5

Component Design and Evaluation

During the design and integration of the RF Channel Emulator, each component, whether implemented in physical hardware, programmable logic, or software, was evaluated and tested. The individual analysis of a component produced data which is used to determine or predict the overall performance of the system. This chapter is devoted to the methods and results of the individual component testing including the main boards used in the final design and the internal logic modules in the DSP path.

5.1 Component Selection

Many of the key components used in the final assembly of the RF Channel Emulator were selected for distinct properties which set them apart from other possible candidates. This section discusses design requirements and the the relevant properties of each of the major components which demonstrate their usefulness in meeting the specifications and requirements of the final design.

5.1.1 FPGA Development Board

The Digilent XUP-V2P board has the features and characteristics needed to implement the digital portions of the design of the RF Channel Emulator. These include the available I/O for communicating to other boards such as the A/D, D/A, and DDS; the powerful FPGA for implementing the required digital logic; the integrated DIMM slot for external RAM; the various I/O ports for Ethernet and RS-232 communication; and the general availability for use in this project. The multiple 40-pin header slots allow for the interfacing of the off board components. An external PC can communicate through the integrated Ethernet board as desired in the project specifications while the RS-232 port allows for ease in debugging with an external PC. The on-board FPGA is a Virtex-2 Pro XC2VP30 FPGA with 30,816 logic cells, 136 18-bit multipliers, 2,448 Kb of BRAM, and 2 PowerPC processors. The integrated DIMM slot allows for up to 2 Gb of RAM which exceeds the required storage capacity needed to implement the specified delay.

The XUP-V2P performs two main functions; it is the link between the other components both internal and external to the RF Channel Emulator, and performs the necessary digital signal processing. It is the former that is to be discussed in this section. The latter function is shown in a simplified block diagram in figure 3.9.

Much of the communication to control the RF Channel Emulator is centered around a soft-core processor located within the integrated FPGA. The processor chosen for this design was the Microblaze. It is the Fast Simplex Link (FSL) of this processor that makes it appealing. This link, or bus, allows for very quick communication between the processor and outside peripherals, but more importantly due to its simple communication scheme, it allows for the very easy design of peripherals to be connected to this bus. These user defined peripherals are the heart of controlling the various DSP elements in the design.

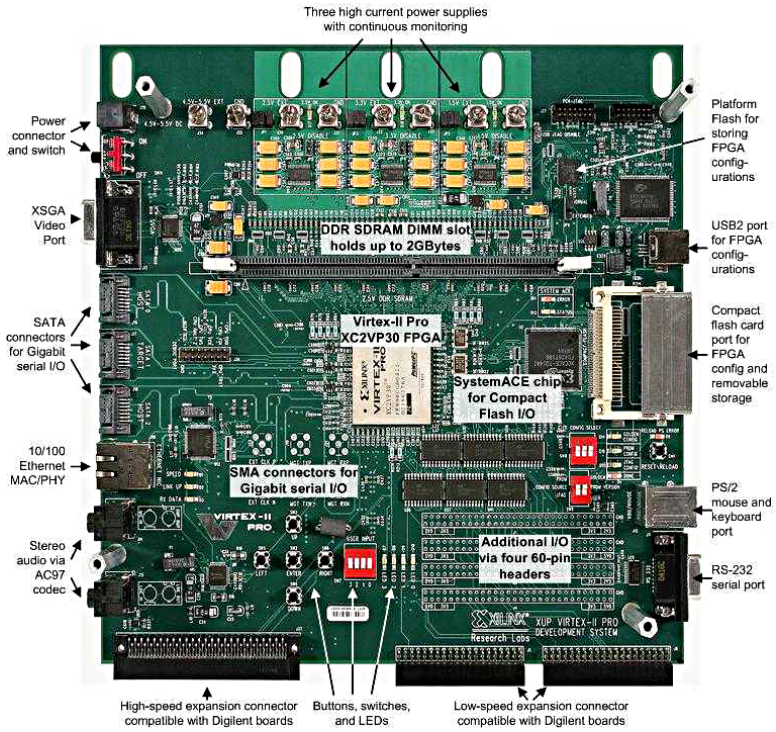


Figure 5.1: Digilent XUP-V2P Development Board.

Figure 5.2: Block Diagram of MicoBlaze and Peripheral Connections.

Aside from the use of the FSL, the On-chip Peripheral Bus (OPB) was used to connect pre-made, drop-in peripherals to the processor. These included the Ethernet PHY, the UART for RS-232 communication, General Purpose I/Os (GPIO) for communicating to external components, and an internal timer. Figure 5.2 shows the basic connection of these peripherals to the internal processor.

Connecting the D/A board and the A/D board made use of the 40-pin header slots mentioned earlier, but since the data was not run through the internal processor but rather through the DSP path, GPIOs were not used for these. Instead, the Virtex series, along with many other FPGAs offered by Xilinx, have primitive hardware structures at the pins themselves called I/O Blocks (IOBs). These IOBs contain flip-flops for both incoming and outgoing data to and from the

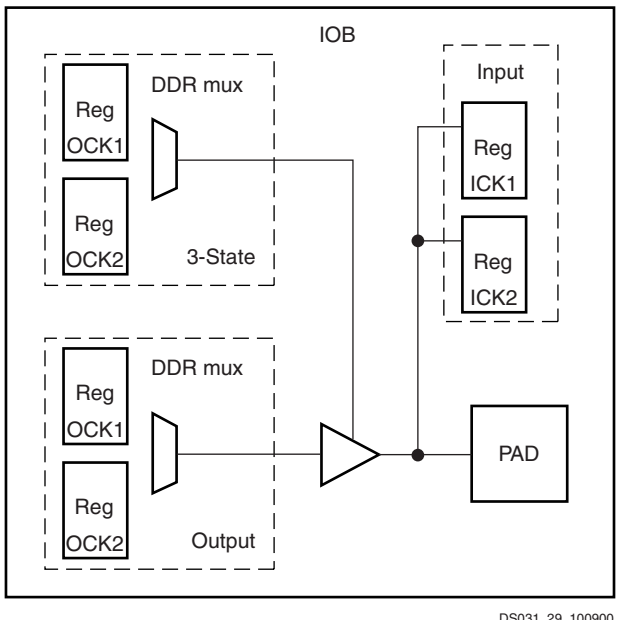


Figure 5.3: Using Input/Output Blocks to Minimize Latency Discrepancies[5]. When flip-flops in IOBs are used discrepancies between latencies are minimized. When they are not used, latencies from pin to pin are uncertain from build to build.

FPGA package pins. This minimizes any internal routing latency discrepancies between data pins which is essential to assure data is clocked out synchronously and in synchronously between corresponding data pins. IOBs are also used for the same reason with the connections to the external DIMM module. Figure 5.3 shows the basics of using the IOBs to ensure minimal latency discrepancies.

5.1.2 Analog to Digital Converter (A/D)

The analog to digital converter chosen for this design was the AD6645. This device is capable of sampling at rates as high as 80 MSPS. As mentioned previously, the bandwidth of the incoming signal is 24 MHz which, according to Nyquist rate derived from basic sampling theory [6], needs to be sampled at least 48 MSPS for proper reconstruction. Our chosen sample rate is 56 MSPS which allows for both proper reconstruction and proper down-converting of the input

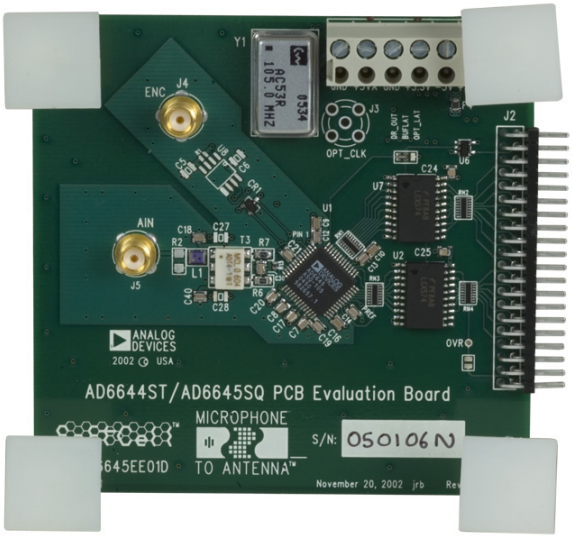


Figure 5.4: Analog Devices AD6645 Evaluation Board.

signal. The AD6645 allows this sampling rate to be achieved. Due to time constraints and the desire for simplicity and modularity, the evaluation board for this chip was purchased to eliminate the need to layout a custom printed circuit board (PCB).

The AD6645 had a listed spurious free dynamic range (SFDR) 100 dBFS and a typical signal-to-noise ratio (SNR) of 75 dB. While the 14-bit resolution of the AD6645 gives it a theoretical dynamic range of 84 dB, the SFDR value shows that quantization noise is spread among the harmonics of the input signal allowing for a lower spurious free range while the SNR shows that the chip introduces other noise aside from harmonic noise as is to be expected with any non-ideal component. The 100 dBFS SFDR is adequate to meet the requirement of 78 dBc spurious free noise. As mentioned in section 3.6, to achieve these values, it is important to have a "clean" clock source. A clock with an unacceptable amount of jitter can dramatically reduce the SNR.

The evaluation board receives its input clock signal through an SMA connector. Data is sent in parallel through shielded ribbon cable to the XUP-V2P.

The clock sent to the AD6645 is split and sent to the Virtex-II chip as well also through an SMA connector to a dedicated clock pin on the FPGA which allows for direct connection to the internal global clock network. This minimizes the internal latency of the clock and data to be in correct phase for proper data capture. Although the 40-pin connector on the evaluation board provided a synchronous clock signal, this was not used due to the lack of drive strength of the on-board buffer to drive a dedicated clock pin on the FPGA, not to mention the 40-pin header on the XUP-V2P board did not tie directly to a global clock input pin. The schematic for the evaluation board was reviewed and showed this synchronous clock pin was, in fact, the input clock signal passed through a buffer. This allowed for the use of external circuitry as described in section 6.1.1.

It should also be noted that the AD6645 produces data in 2's compliment form. This allows for ease of use when processing this data through the DSP path.

5.1.3 Digital to Analog Converter (D/A)

Although the data path inside the FPGA has a center frequency of 14 MHz and a bandwidth of 24 MHz, the data output to the D/A is upconverted to have a center frequency of 42 MHz keeping the same bandwidth but having a sample rate of 112 MSPS. To accurately reconstruct this signal, the AD9772A chip was chosen. It maintains the 14-bit resolution of the AD6645 but can handle sample rates of up to 160 MSPS. For the same reasons that the evaluation board was chosen for the A/D component, the D/A component was also purchased on an evaluation board.

The AD9772A is rated as having a spurious free dynamic range of 74 dB. This value is close to the desired 78 dB spurious free range. As mentioned in section 3.5, care has been taken to reduce this value even further in most operational cases.

The AD9772A has an internal 2x interpolator which simply doubles the output

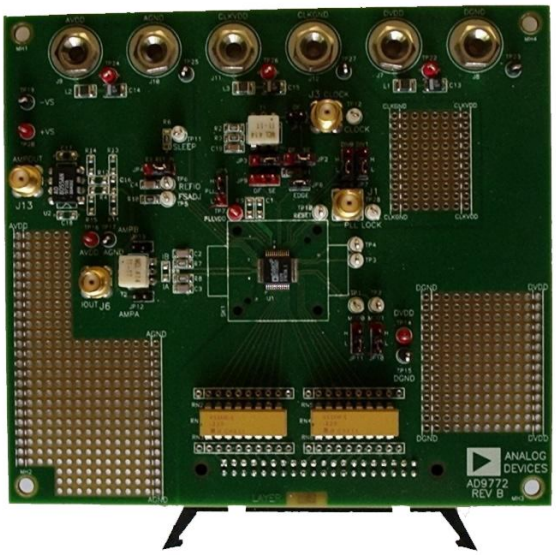


Figure 5.5: Analog Devices AD9772A Evaluation Board.

data rate of the incoming data, inserts zeros between the samples, and passes the data through a lowpass filter. With an simple jumper setting on the evaluation board, a highpass filter can be selected instead of the lowpass. This allows the final upconversion from a center frequency of 42 MHz to the desired value of 56 MHz without the addition of an external circuit. This not only reduces spurious noise as mentioned in section 3.5 but also reduces the energy in the spectral images. Figure 3.17 shows this process.

As with the A/D evaluation board, the AD9772A is connected to the XUP-V2P via a shielded ribbon cable. Also in the same vein as the AD6645, the SNR of the AD9772A can be dramatically decreased using a clock signal with an unacceptable amount of jitter. Section 3.6 describes this in more detail, and the oscillator section shows experimental evidence supporting this issue.

5.1.4 Oscillator

As previously mentioned, the DDS is used to create the sampling clock of 56 MHz. The DDS requires a reference oscillator to produce the desired frequency

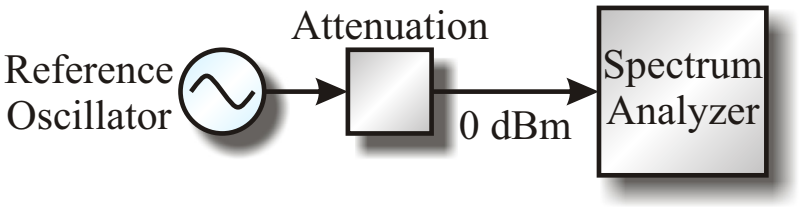


Figure 5.6: Test Setup for Measuring the Spectral Plots and Phase Noise of the Reference Oscillators.

output. Any jitter on the reference clock is transposed to the produced sinusoid, thus a very "clean" reference clock is desired. A "clean" clock has low jitter, or in other terms, low phase noise as well as has low deterministic, or spurious, jitter or phase noise as well.

Initially, the primary reference oscillator selected for the DDS was the Crystek CPR033-100. This clock is rated as having a typical jitter of around 0.5 ps, a maximum RMS jitter of 1 ps, and a maximum frequency stability of ± 25 ppm from 0°C to +70°C. The output power of this oscillator is around 13 dBm. To measure the output spectral plot, a 13 dB attenuator was applied to the spectrum analyzer input. Figure 5.6 shows this test setup. Figure 5.7 shows the spectral plot of the CPR033-100.

Phase noise was measured using the Agilent E4445A Spectrum Analyzer as was the spectral plot. This instrumentation is capable of measuring phase noise directly without going through a lengthy process of measuring each frequency offset point with a resolution bandwidth of 1 Hz. Phase noise itself is relative energy per Hz of a particular offset frequency in comparison to a center frequency.

Figure 5.8 shows the measured phase noise of the CPR033-100 oscillator labeled the 100 MHz clock. Through a simple conversion [15] the phase noise can be converted to jitter. This too is shown in figure 5.8. As one can see, the value measured, 291.13 ps, is much higher than the listed typical value of 0.5 ps. The limit of what the instrumentation is able to measure has been reached. Also, it

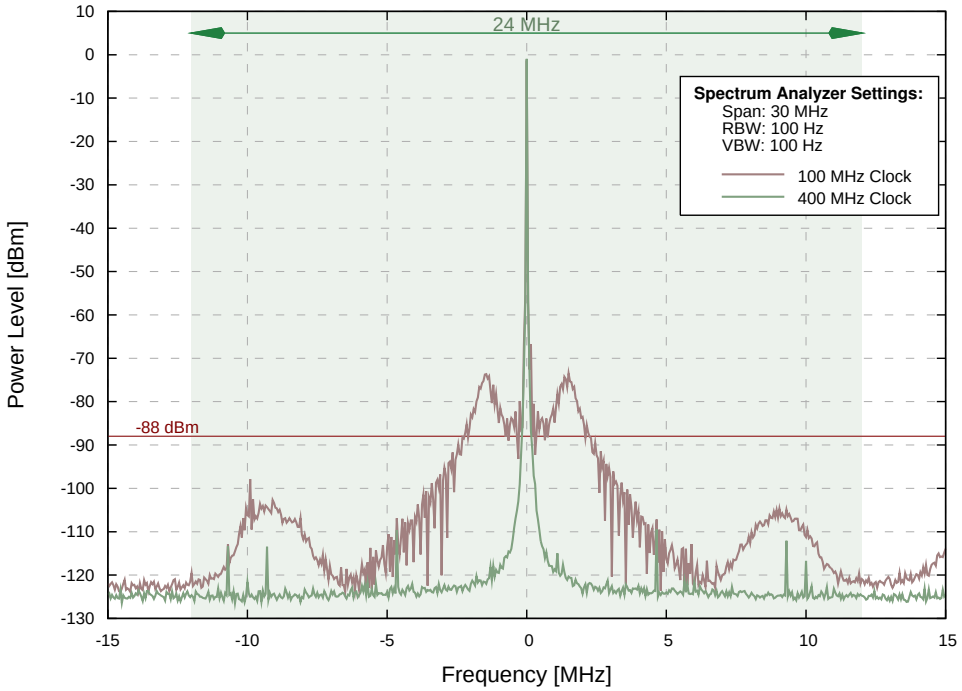


Figure 5.7: Spectral Plots for Oscillators.

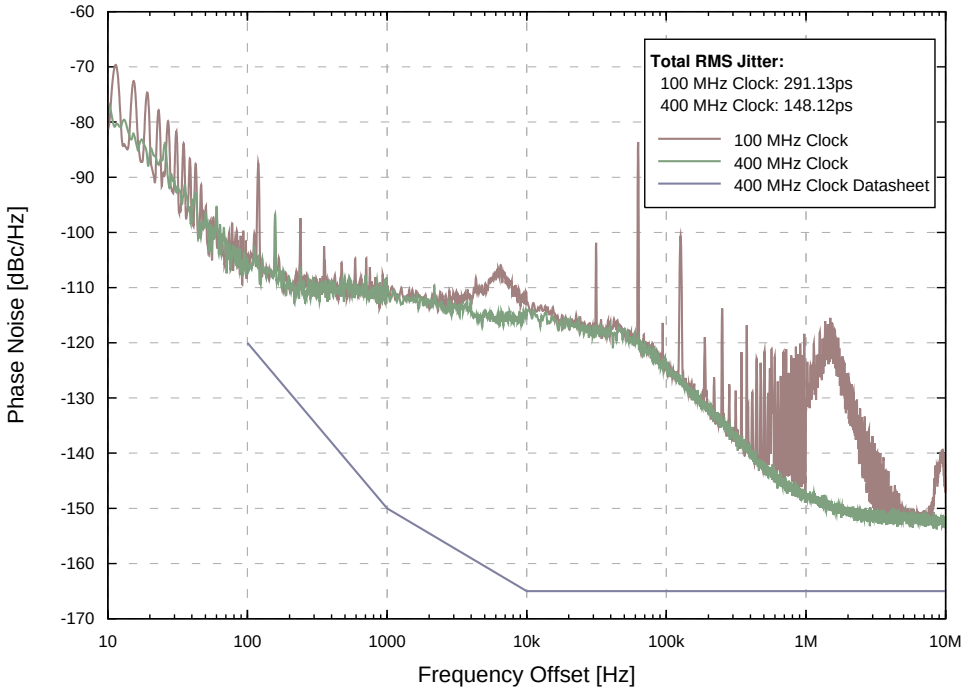


Figure 5.8: Phase Noise Plots for Oscillators.

should be noted that more than just phase noise is being measured here. These topics are further discussed in section 7.1.1.

The spectral plot of the 100 MHz oscillator along with the fact that the noisy internal PLL of the DDS was needed to multiply an internal frequency to 400 MHz made it clear that a "cleaner" oscillator was needed. For this, the Wenzel Sprinter series was chosen with a frequency of 400 MHz such that the PPL of the DDS could be avoided. This has a rated jitter of less than 0.18 ps–significantly less than that of the original oscillator. Figures 5.7 and 5.8 show the results of the Wenzel part. The improved spectral plot is dramatic while the phase noise plot shows some improvement although still limited by the instrumentation. The Wenzel part shows an improved measured jitter of 148.12 ps; this still falls far short of the the specifications. The Wenzel part does have some specifications on what the expected phase noise should be. As one can see in figure 5.8 these results are unobtainable with the Agilent instrumentation used.

5.1.5 Direct Digital Synthesis

This section focuses on the component and configuration choices in determining the final configuration of the DDS as the clock source. Already mentioned were the requirements for a clock with very low stochastic and deterministic jitter as the source of the reference clock for the DDS. The effect of the two oscillators on the final performance of the A/D and the D/A as well as in a pass through configuration (sinusoidal injection test) are discussed.

Clock Synthesis with the CPR033-100

Initially, the 100 MHz oscillator (CPR033-100) was used to drive the DDS with an internal clock multiplier of 4x to achieve the 400 MHz clock from which the 56 MHz clock was generated. The spectral plot and phase noise plots can be seen in figure 5.9 and figure 5.10 respectively. As one can see, the spectral image

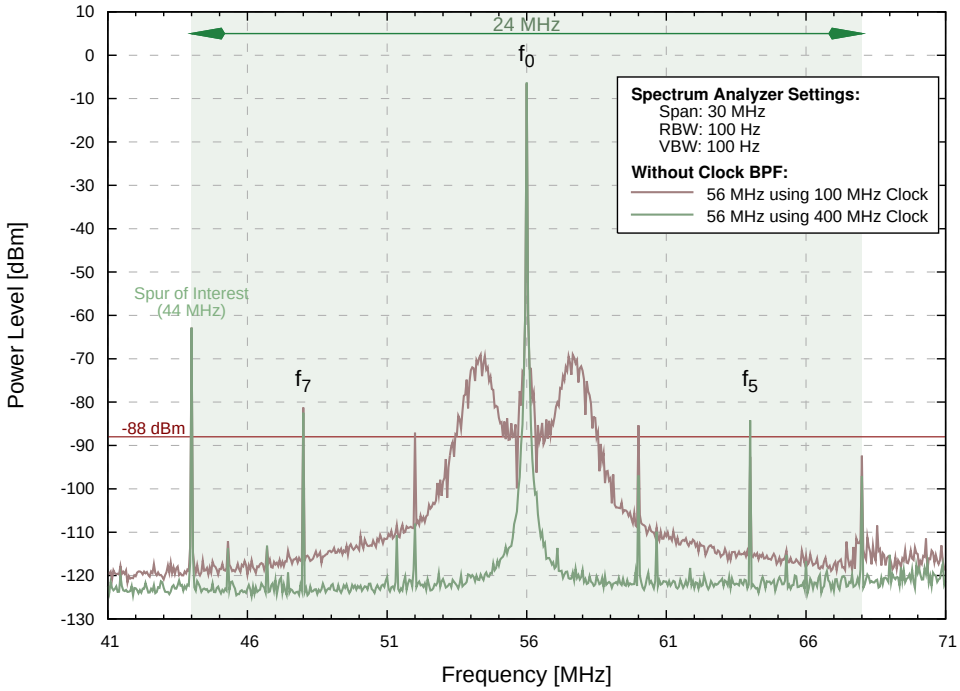


Figure 5.9: Spectral Plot for 56 MHz Clock Generated with the DDS.

of the input clock has been convolved with the output sinusoid at 56 MHz. Aside from that, additional spurs have been introduced. At first glance, these spurs, given their fairly periodic nature, seem to fall on the harmonics that would be generated due to quantization noise of the 14-bit internal D/A converter inside the DDS. These spurs are clearer on the 400 MHz oscillator discussed below.

Clock Synthesis with the 400 MHz Oscillator

Using the 400 MHz, Wenzel oscillator as a reference dramatically reduces all noise in general out of the DDS. For the most part, the spurious noise introduced has the same magnitudes and frequencies as those introduced using the 100 MHz reference oscillator. This indicates that they are a construct of the DDS. Again, their frequencies appear to be quantization harmonics due to their relative frequencies, but when analyzed, only two fall on actual harmonics. The fundamental frequency is labeled f_0 while the two harmonics are designated by

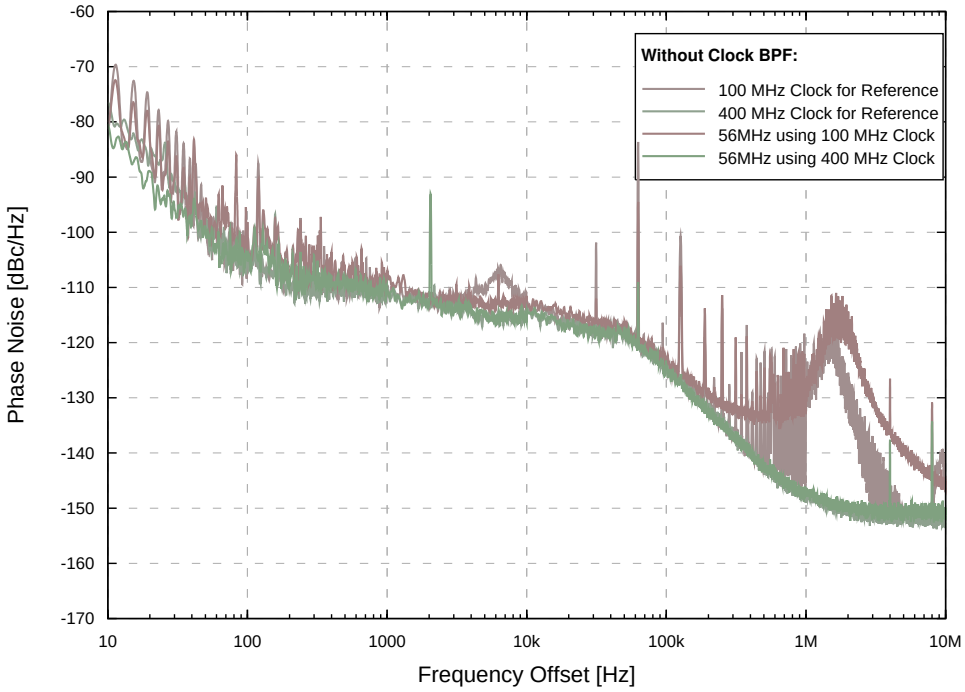


Figure 5.10: Phase Noise Plot for 56 MHz Clock Generated with the DDS. The noise floor of the spectrum analyzer dominates much of the phase noise for much of the frequency range in each measurement.

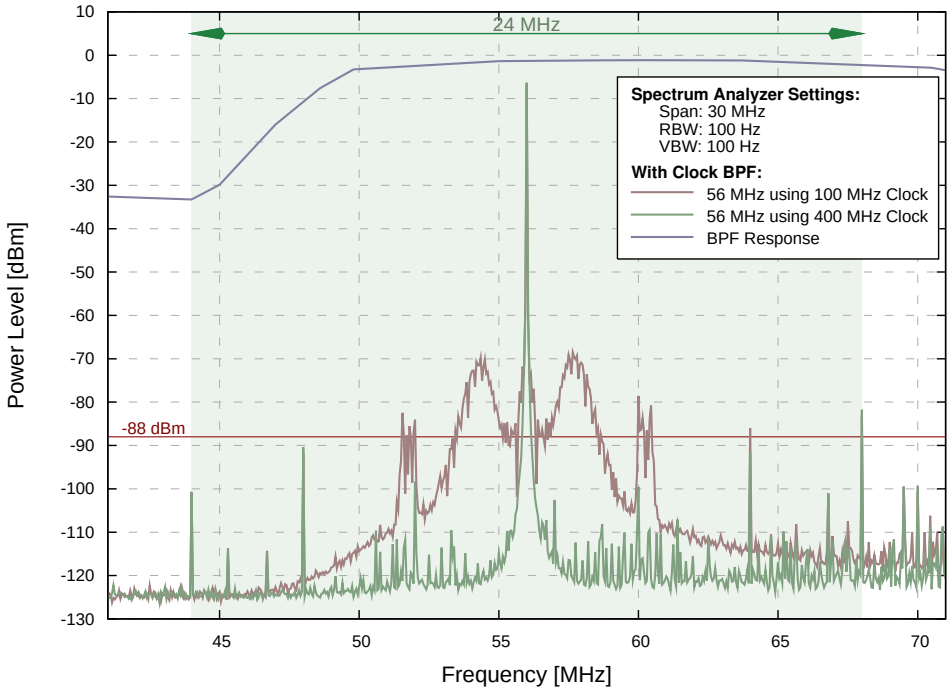


Figure 5.11: Spectral Plot for 56 MHz Clock Generated with BPFs on Clocks.

their harmonic number, namely f_5 and f_7. The source of the other anomalous peaks is unknown aside from the fact that they are artifacts of the DDS itself. This is discussed in greater detail in section 7.1.2.

One spur of interest, labeled so in figure 5.9, is around 44 MHz. This peak has a magnitude high enough that it introduces unwanted spurious peaks in any sampled data. This can be seen in subsequent sections. For this reason, an off the shelf bandpass filter is used that attenuates this particular spur.

Figure 5.11 shows the spectral plot of the clock signal with the introduction of the bandpass filter along with the response of the bandpass filter. As would be expected, the unwanted spur is attenuated. Section 7.1.3 demonstrates the effect of sampling with and without this spur. For comparison, figure 5.12 shows the phase noise plot of the filtered clock.

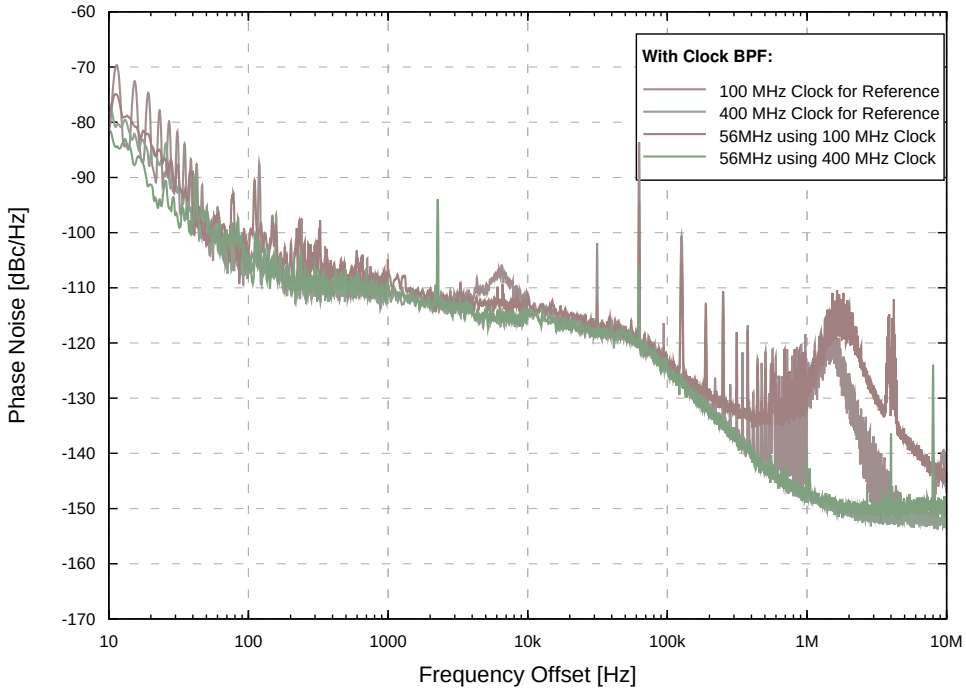


Figure 5.12: Phase Noise Plot for 56 MHz Clock Generated with BPFs on Clocks.

A/D Performance with Various Clock Configurations

To demonstrate the performance of the analog to digital converter with the various clock configurations, a sinusoid is injected into the system at -10 dBm (the specified input power level for the system), digitized by the A/D, and ran through a fast Fourier transform (FFT) to show the spectral result. In order to achieve a decent resolution, 8192 samples are taken for the FFT. A Blackman window is used to reduce spectral leakage with the side effect of also reducing spectral resolution. It should be noted that for all the data taken in this section, the filtered clock is used as it is in the final design as is shown in figure 5.13.

Figure 5.14 shows an injection frequency of 65.33 MHz. Aside from out of the band of interest noise, the only peak above the desired noise threshold aside from the fundamental is the first harmonic which sits at approximately -60 dBc. This is definitely lower than what is specified in the data sheet; one must consider

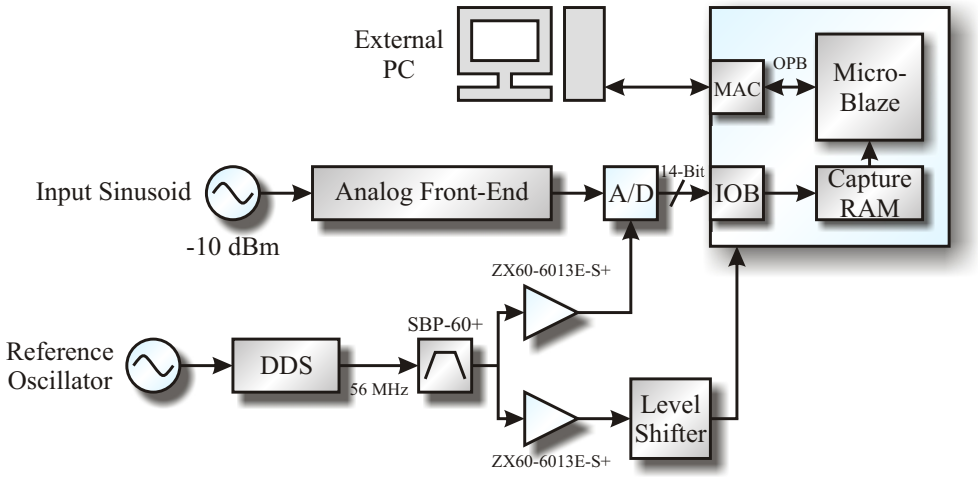


Figure 5.13: Setup for Measuring A/D Performance Given Different Reference Oscillators.

that the -100 dBFS was, by definition, at full scale. Here, some head way is left to assure that we do not overload the A/D. Another discrepancy of note is that this setup includes the analog front end to accurately gauge the performance of the system up to the A/D. The inclusion of any other components will definitely reduce the overall performance.

The spurious free dynamic range of the system using the 100 MHz clock sits at the same value of -60 dB, although it is definitely apparent that there is dramatically more in-band spurious noise.

Figure 5.15 shows an injection at 70 MHz. This should produce the "cleanest" results given that all harmonics should fall at the injection frequency or out of the band of interest. This appears to hold true for the most part with the 400 MHz clock. A single spur can be seen to breach the noise threshold; given its location at approximately 63 MHz, it could be attributed to induced noise. The signal sampled with the 100 MHz clock has noticeably more spurious noise above the noise threshold.

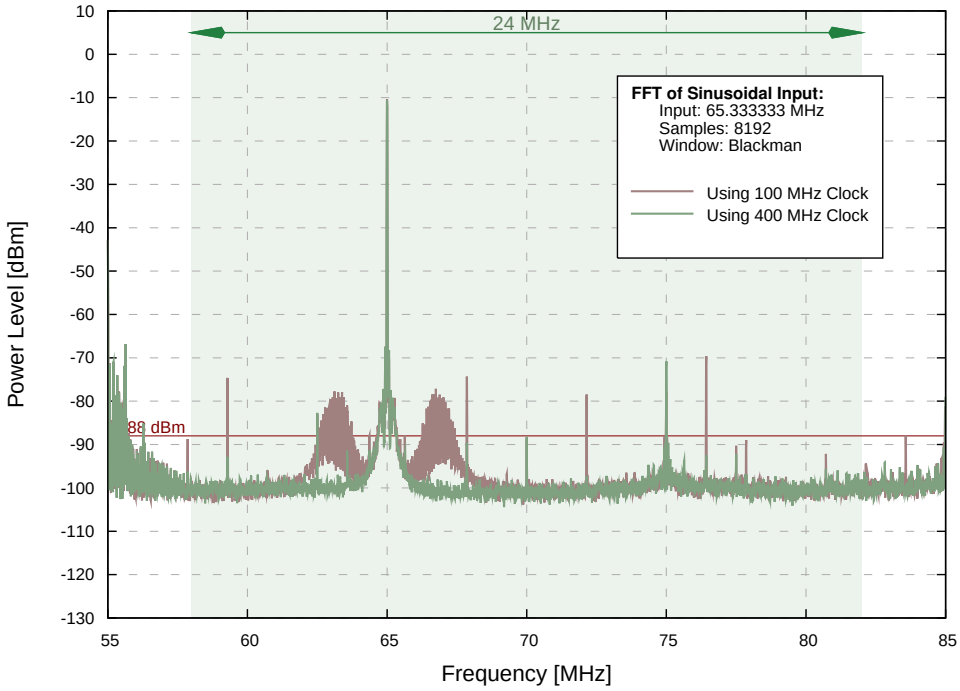


Figure 5.14: FFT of A/D with a $\frac{7}{6}f_s$ (65.33 MHz) Input.

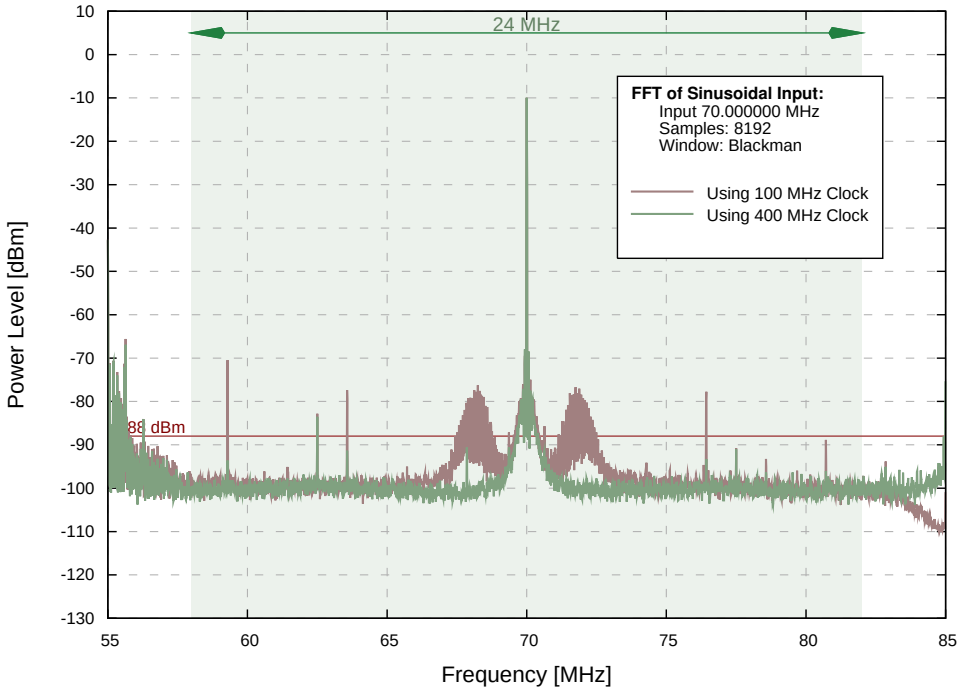


Figure 5.15: FFT of A/D with a $\frac{5}{4}f_s$ (70.0 MHz) Input.

D/A Performance with Various Clock Configurations

To demonstrate the performance of the digital to analog converter given various reference oscillators driving the DDS, the setup in figure 5.16 was used. An external PC was used to send commands to the embedded Microblaze processor which set the desired output frequency of the sinusoid generator. The output of the sinusoid generator was sent directly to the D/A. The sinusoid generator functions from the same 56 MHz source clock that is used to clock in the A/D data. The clock used to drive the output data to the D/A is taken from the D/A itself to ensure synchronization. The internal clock of the D/A is driven at 224 MHz, thus, given that there is an internal x2 interpolator, the D/A clocks in data at 112 MHz. The 112 MHz clock from the D/A to the FPGA is passed through a digital clock manager (DCM) upon entry; from this DCM, a 56 MHz clock is extracted to function with the clock used to clock in the A/D data. Having the two clocks at the same frequency allows for simple logic to be used along the DSP path to account for any phase difference between them. In order to increase the data rate back to 112 MHz, a double data rate (DDR) input/output block (IOB) is used along with an upconverter. This is discussed in more detail in section 5.2.5.

Figure 5.17 shows the spectral plot of the output with a generated output sinusoid at 65.33 MHz with a power level of -10 dBm to best demonstrate the performance of the D/A under normal operating conditions. It can be seen that using the 400 MHz clock, the only spur above the noise threshold is the first harmonic, upconverted, of the output frequency. This makes the spurious free dynamic range of the D/A to be about 68 dB. Although this is closer to the value listed in the data sheet than was the A/D, performance is still slightly lower than expected. This could be due, in part, to the quantization error of the sinusoidal generator. This error is discussed in more detail in section 5.2.3. The use of the 100 MHz reference clock, while showing a significantly worse SNR, has a SFDR of roughly the same as the 400 MHz reference clock.

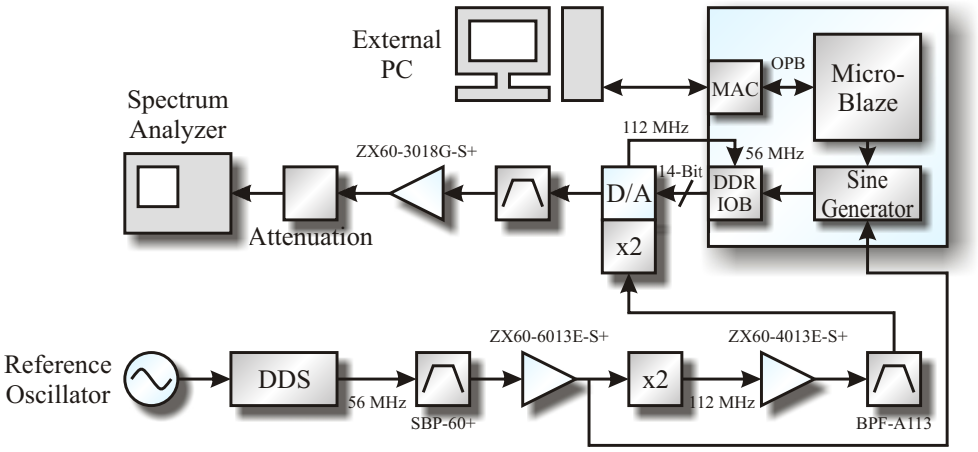


Figure 5.16: Setup for Measuring D/A Performance Given Different Reference Oscillators

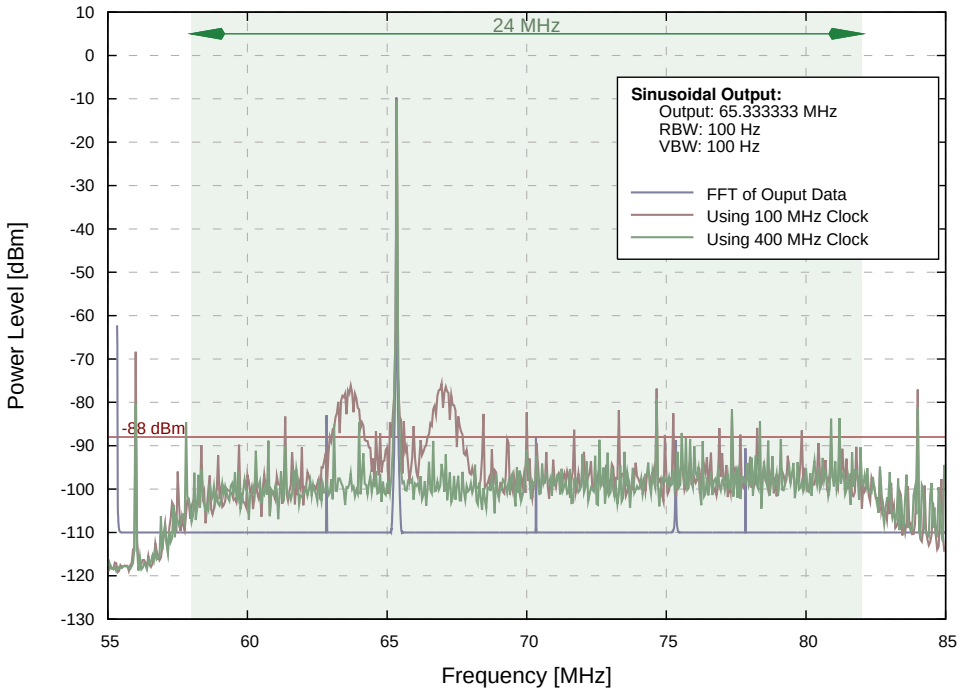


Figure 5.17: Spectral Plot of D/A with a $\frac{7}{6}f_s$ (65.33 MHz) Input.

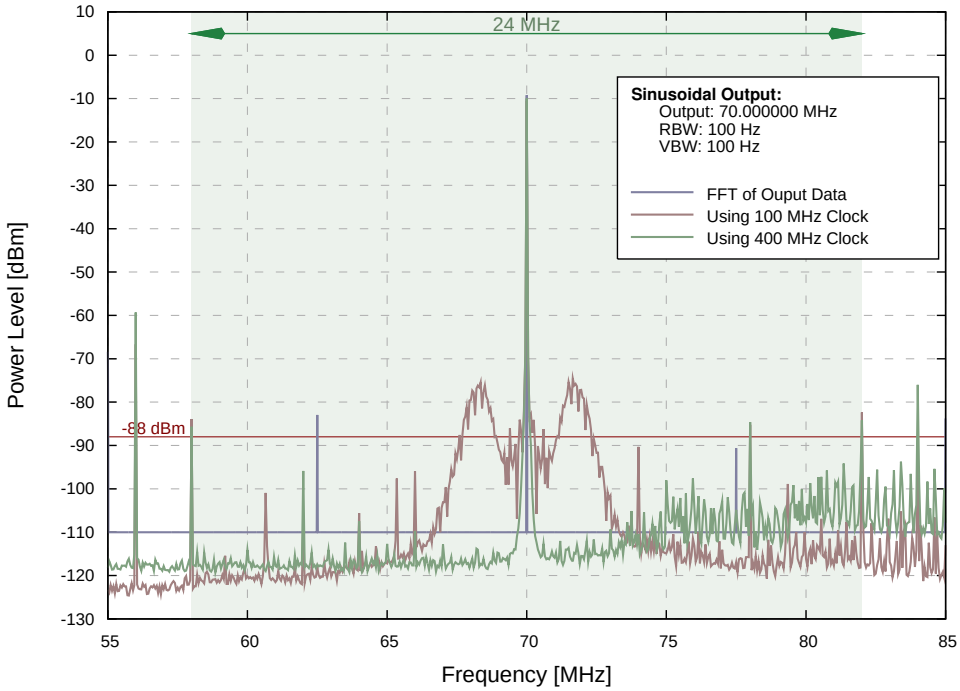


Figure 5.18: FFT of D/A with a $\frac{5}{4}f_s$ (70.0 MHz) Input.

Figure 5.18 shows the spectral plot of the output with a generated output sinusoid at 70 MHz again with a -10 dBm power level. This frequency should show the best case scenario given that all harmonics fall either out of the band of interest or on the center frequency. Using the 400 MHz reference clock, the spurs show a pattern similar to the output of the 56 MHz sampling clock generated by the DDS. The SFDR here can be seen to be 78 dB for in-band spurs. The 100 MHz reference clock, still maintains a worse SNR without a significant change in the SFDR which stayed at about 65 dB.

Sinusoidal Injection Performance

To demonstrate the performance of the system including both the analog to digital converter and the digital to analog converter, the setup in figure 5.19 was used. Here, an input signal generated by a function generator was used to input a signal at the desired nominal input value of -10 dBm. This signal was conditioned

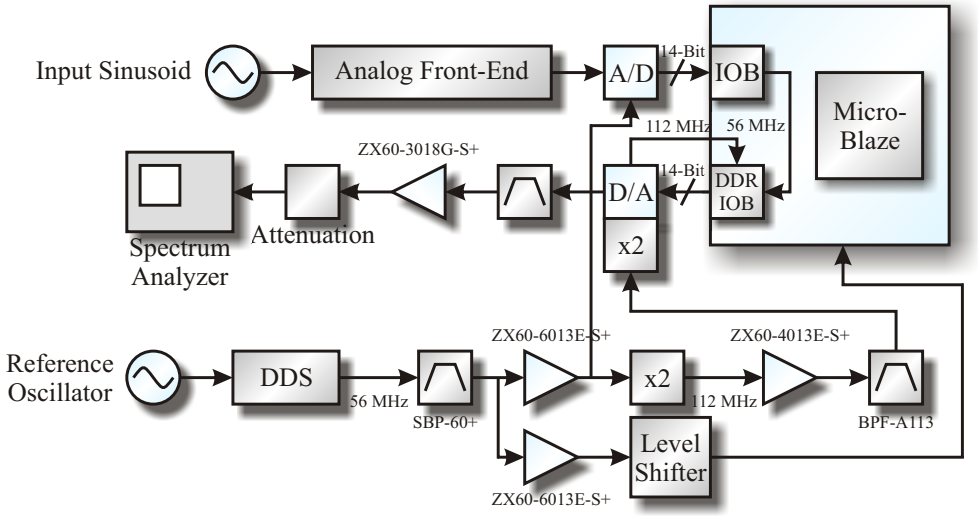


Figure 5.19: Setup for Measuring System Performance Given Different Reference Oscillators.

through the analog input stage, digitized by the A/D, passed directly to through the FPGA to the D/A, and then analyzed on a spectrum analyzer. Again, both the 100 MHz and the 400 MHz reference oscillators were used as a comparison.

Figure 5.20 shows the output of this setup with an input signal at 65.33 MHz and -10 dBm. The results of this figure are fairly similar to those obtained by the A/D and the D/A test. The 400 MHz and 100 MHz reference clocks had spurious free dynamic ranges of roughly 65 dBc, while the signal to noise ratio of the 100 MHz reference clock was worse than that of the 400 MHz clock.

Figure 5.21 shows the results of a 70 MHz, -10 dBm input signal. Again, this setup shows the best case scenario with harmonics falling outside the band of interest or at the center frequency. Here, the effects of the spurs in the sample clock generated by the DDS become more apparent as with the D/A testing. The in-band SFDR of the 400 MHz is approximately 77 dB–very close to the desired value of 78 dB. The 100 MHz signal SFDR stayed at 65 dB. The SNR for the 100 MHz reference clock was significantly worse than the 400 MHz reference clock with the continued persistence of the side lobes introduced by the reference clock

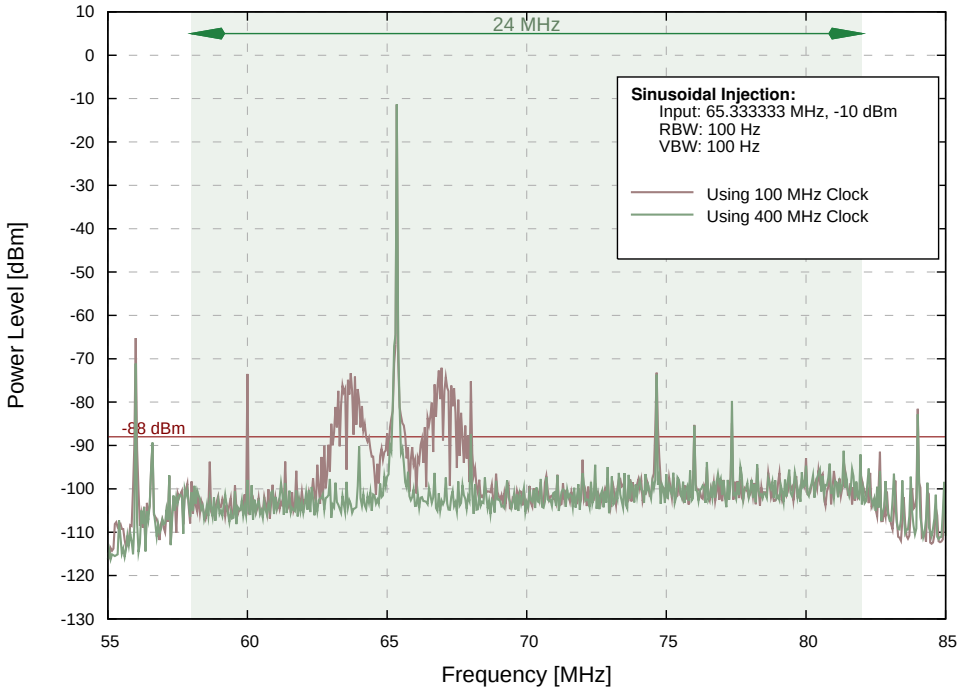


Figure 5.20: Spectral Plot with a $\frac{7}{6}f_s$ (65.33 MHz) Input.

itself.

For strictly comparison purposes, the phase noise measurements are included in figure 5.22. As mentioned previously, these measurements not only include noise aside from phase noise, but also are limited by the instrumentation from which they were produced.

The injection test on the initial design performed by Woolrich [22] used a 70 MHz input signal at -10 dBm. The initial design had the following configuration: the 100 MHz reference clock was used; the internal PLL of the A/D was used for clock multiplication; the internal PLL of the D/A was used for clock multiplication; and the analog output stage used a mixer introducing another stage of phase noise into the signal. The analog input stage stayed untouched. Figure 5.23 shows a comparison of the initial design along with the final design with and without the bandpass filters on the clocks.

The results, given the different resolution windows used in the measurements,

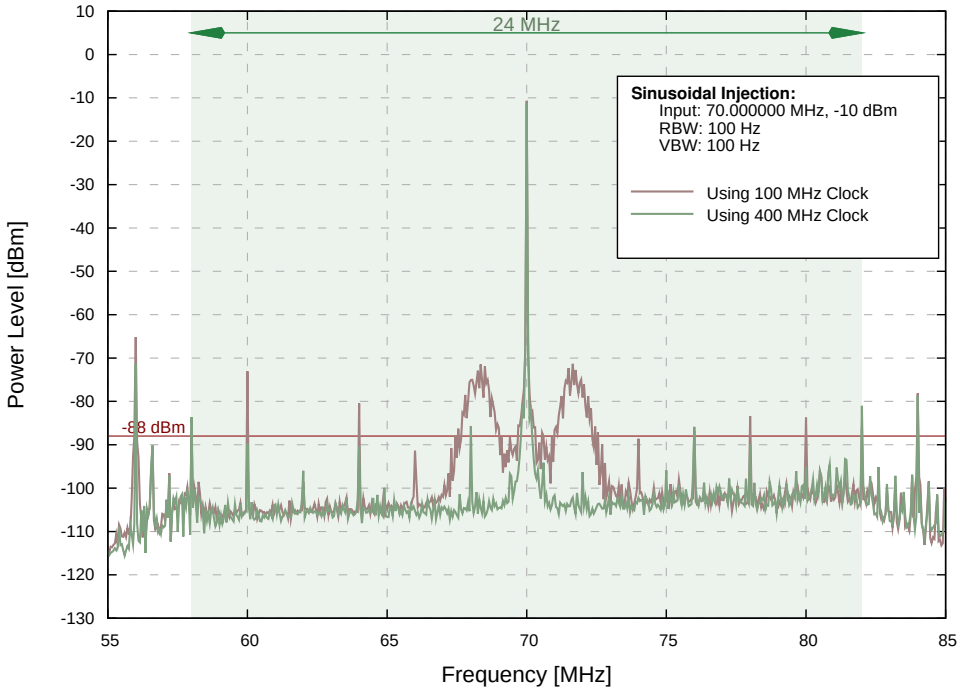


Figure 5.21: Spectral Plot with a $\frac{5}{4}f_s$ (70.0 MHz) Input.

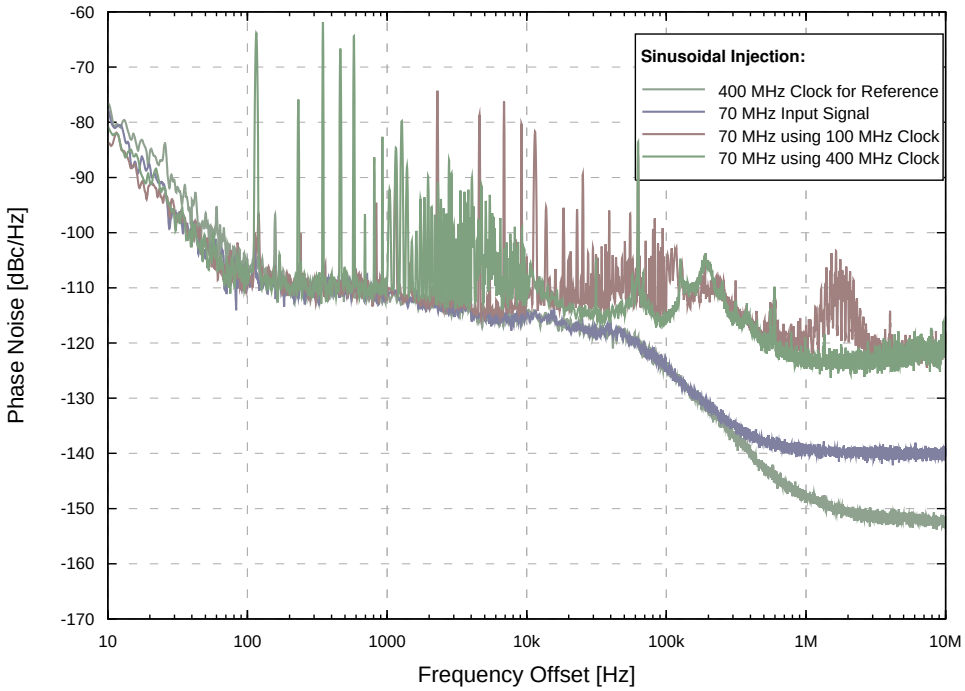


Figure 5.22: Phase Noise for Injection Test.

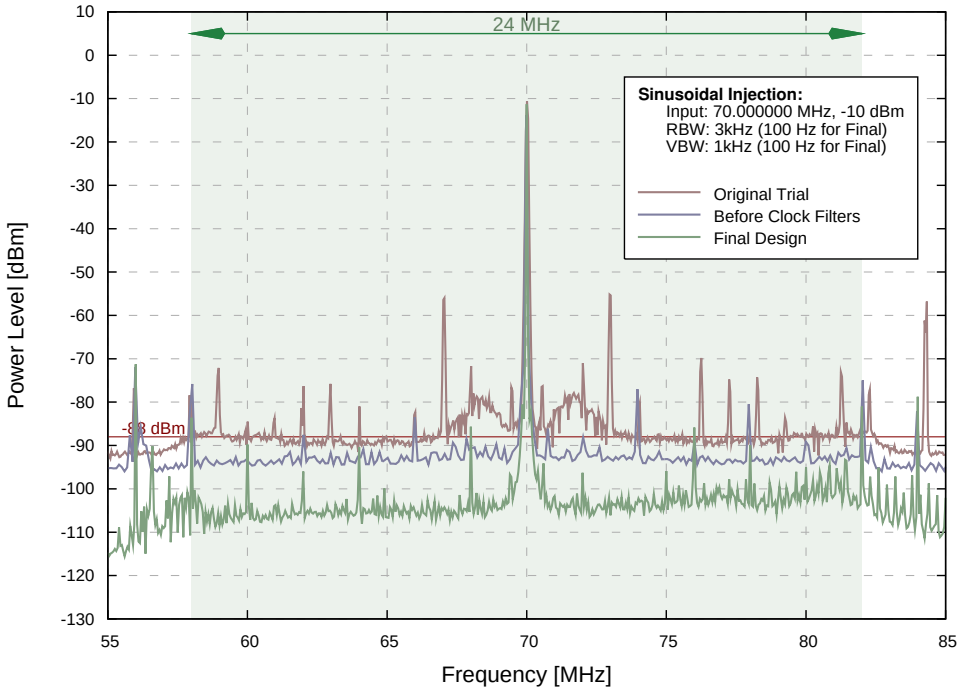


Figure 5.23: Spectral Plot of Various Designs with 70 MHz Injection.

can only accurately show the spurious free dynamic ranges of each design. The initial design shows a SFDR of roughly 45 dB, while the design without bandpass filters on the sampling clocks shows a SFDR of roughly 68 dB. The final design maintains its SFDR of 77 dB, a significant improvement over the other designs.

5.2 Digital Components

This section focuses on the design and implementation of the components which reside in the logic of the FPGA. The first of these is the memory controller which is used to handle the coarser digital delay functionality through the use of an external DDR SDRAM DIMM. The next is the dynamic interpolation component which is used to handle both the fine delay settings and fine, dynamic delay changes used in the implementation of the Doppler Effect. The next is the single side band modulation component also used in the implementation of the

Doppler Effect. After this follows the attenuation and additive white Gaussian noise components. The last digital component in the DSP path is the output filter and upsampler which correct for any analog component deviations from a flat response in the band of interest and upconvert the signal to allow the D/A to be used for the final up conversion respectively. There is a final component that is used in multiple places along the DSP path to monitor the power levels.

5.2.1 The Memory Controller

The memory controller acts as a large FIFO as described in section 3.4.1. By having the ability to set the spacing between the address at which the

incoming data is written to and the address at which the outgoing data is read from, the delay can be dynamically changed. The delay component is placed first in the DSP path since any other changes to the system parameters are expected to be as near instantaneous as possible. The design presented in this section allows for this spacing to be set to a granularity of one sample. Any finer delay settings are handled with the interpolator described in more detail in the next section.

The basic function of the memory controller is shown in a very simplified state diagram in figure 5.24. At power up, the external DIMM must be initialized according to the specifications in the data sheet. This is the task of the first state. Internal to this state, the controller runs through a list of initialization commands which it sends to the memory module.

Once the initialization is complete, the memory controller moves into an idle state where it waits for a signal from the read or write stages. These two stages are shown in figure 5.25 and figure 5.26. When a need to read or need to write signal is triggered, the controller performs the appropriate task (writes have priority if both are triggered within the same clock cycle). Both the read and the write states take 32 clock cycles to complete before returning back to the idle state.

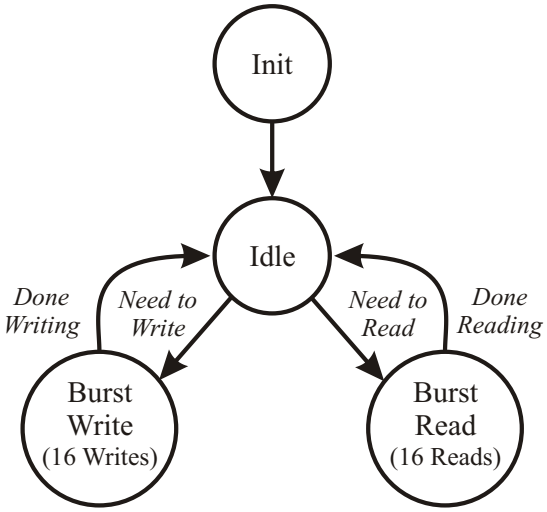


Figure 5.24: Memory Controller Simple State Diagram.

16 of these clock cycles are to initialize the appropriate location in the external memory module for the desired transaction. Then, a 16 cycle read or write burst is executed.

Calculating the throughput of the memory controller requires a basic understanding of the external memory module as well as some caveats of the controller itself. The DIMM has a 64-bit data bus allowing for four 16-bit samples to be accessed simultaneously. Due to the dual data rate nature of the memory, two sets of four samples are accessed with each clock cycle. With a burst of 16 clock cycles, 128 samples can be accessed per burst. A read transaction has a delay of three additional clock cycles due to the read latency inherent in the SDRAM. Since throughput requires both a read and a write as well as an extra clock cycle for a transition between them, it can be said that 128 samples are read and written in 68 clock cycles (32 for writing, 35 for reading, and one for transition). The memory controller runs at the internal clock frequency of 100 MHz; this means that the final throughput is 128Samples $\times$ 100MHz/68clocks $\approx$ 188MSPS. This greatly exceeds the system requirement of $56MSPS$.

There are quite a few different ways in which the throughput could be in-

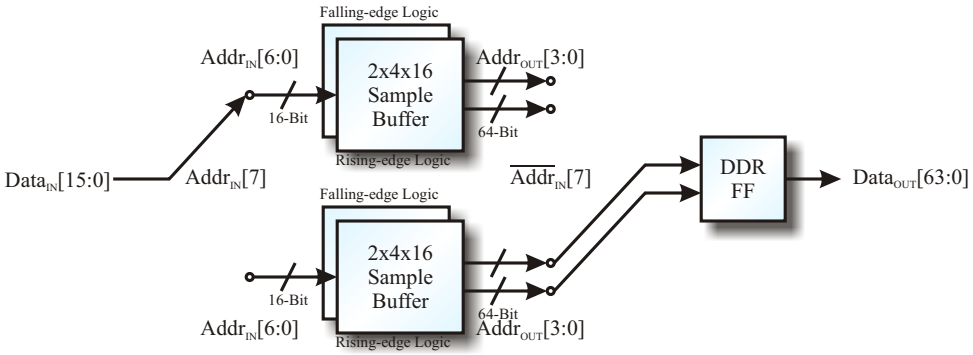


Figure 5.25: Memory Controller Write Logic.

creased. For one, the 16 cycles used to initialize the appropriate location in the external memory module are not always required. This represents what is needed in the worst case assuming that the banks need to change and all addresses are precharged since the last burst transaction. Always assuming the worst case allows for a simpler implementation which translates to less logic used with the trade off of a design with slightly less throughput. Also, the burst size can change the throughput with similar trade-offs. The burst size, according to the DDR SDRAM data sheet, allows for up to 128 clock cycles or 256 64-bit transactions. This would mean less proportional time spent preparing for the transaction than executing the transaction. Changing the burst size would not only increase the needed internal memory caches (increase in logic in the FPGA), but would also mean an increase in the minimum delay since a burst write is required before a burst read if all data fed to the DSP path comes from the external memory. Such a topology is beneficial for a simplified design given that additional logic would be needed to bypass the external memory for shorter delays.

The logic that handles the writes to the memory controller is shown in a simplified block diagram in figure 5.25. The core of this logic is comprised of FIFOs that hold 16 samples. This size was chosen to easily fit the Xilinx primitive for asynchronous two port FIFOs that allow for reading and writing from two different locations at two different clock frequencies. This allows data to be

clocked in at 56 MHz while clocked out at 100 MHz. These primitives are what Xilinx refers to as distributed RAM (DRAM). This is in contrast to the much larger block RAM (BRAM) primitives in the Virtex and other Xilinx FPGA families. Using the DRAMs allows the size of the end FIFOs to be user selectable without wasting logic as opposed to using BRAMs which are a fixed size. They also make meeting the internal timing requirements, such as setup and hold times, easier to meet since the DRAMs can be placed in an available slice.

The DRAM primitive inside the Xilinx FPGAs takes in a four bit address while storing a single bit in each location. This allows the data width of the final FIFO to be size selectable by the number of DRAMs used. In this design, a total of eight of these DRAMs were used for addressing the 16-bit wide input data. This makes a total of 128 samples that can be stored: the amount needed for a single burst write operation as described above.

Two of these 128 sample buffers are used in the design. This makes a total of 256 samples that can be stored at one addressed as shown in figure 5.25 as $\text{Addr}_{\text{IN}}[7:0]$. Bit $\text{Addr}_{\text{IN}}[7]$ selects between the two 128 sample buffers. When this bit changes value, this signals that the last buffer has been filled and is ready to be written to the external ram. Thus, the 'Need to Write' flag is set. Given that the theoretical throughput is over double the necessary throughput, the second 128 sample should never fill without the first being written out to external memory eliminating the possibility of buffer overrun.

As mentioned before, the DRAMs, being comprised of single bit buffers, allows the data bus input and output width to be defined as needed. Here, the input data width is comprised of eight 16-bit data samples. To accommodate the 64-bit data bus of the external DDR memory, the output width is designated as two 64-bit data blocks: one to be clocked out on the rising edge and one to be clocked out on the falling edge. To accomplish the same thing using BRAM, at least eight BRAMs would be needed which would make the FIFO much larger than needed even for the maximum bust length that the external memory can handle.

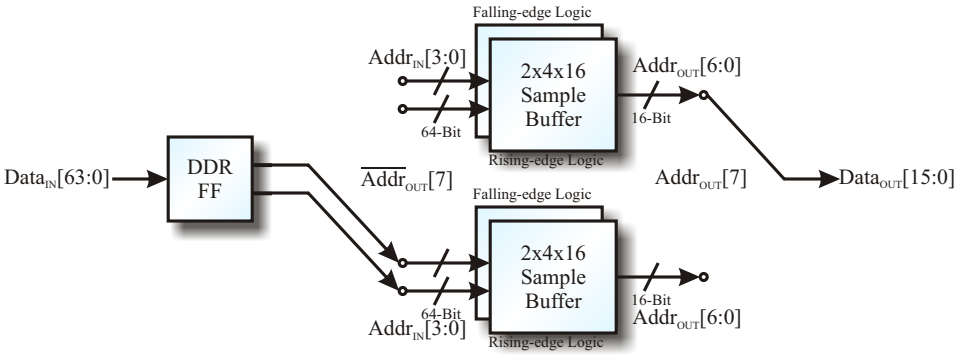


Figure 5.26: Memory Controller Read Logic.

The logic to handle reading from the external memory is shown in the simplified block diagram in figure 5.26. This logic and design is almost identical to the write logic except with reverse flow. As data is read from the two 128 sample FIFOs and $\text{Addr}_{\text{OUT}}[7]$ changes, the memory controller is flagged by the 'Need to Read' signal which then fills the 128 sample buffer that currently isn't being addressed with the next block of data. This means that there is always data ready in the FIFO. This meets the one specification in section 3.4.1 that requires that the next data sample be always available for the interpolator.

With the size of the FIFOs as they are, the minimum delay can be empirically calculated. This will be the time it takes to receive enough data to have it written to the external memory, have that portion of memory read back, and then the portion of the internal FIFO be reached by the read logic. That comes to a total of 256 samples or roughly 4.57μsec. This is significantly less than the 1 ms design goal.

5.2.2 Interpolation Finite Impulse Response (FIR) Filter

The design guidelines of the interpolation filter are discussed at length in section 3.4.2. A sinc interpolation scheme has been implemented here which requires the real-time calculation of many points along a sinc curve. Using a

table look-up for these points along with a linear interpolation allows for a fairly high degree of resolution that can be attained while maintaining a high level of accuracy. As previously mentioned, the linear approximation, although requiring more logic, reduces the size of the tables fairly significantly. This allows the use of the block RAM (BRAM) of the Virtex series to be utilized as look-up tables. Taking advantage of the even nature of the sinc function along with the dual-port nature of the BRAM, the table size is cut in half. This approach is discussed in detail in the following sections.

Coefficient Generation

Generating the coefficients for the finite impulse response filter used to interpolate between samples is designed around the sinc interpolator approach. The equation for the continuous time signal reconstructed from the individual samples, shown in figure 3.6, can be rewritten in terms of a sample, n, and a deviation from that sample, Δn.

$$\begin{aligned} x_c(t) &= \sum_{k=-\infty}^{\infty} x_s[k]\text{sinc}[(t - \frac{k}{f_s})f_s] \\ x_c(n + \Delta n) &= \sum_{k=-\infty}^{\infty} x_s[k]\text{sinc}[(n + \Delta n - \frac{k}{f_s})f_s] \end{aligned} \tag{5.1}$$

The interpolator never needs to be outside of the range $\pm\frac{1}{2}$ sample. This is due to the fact that any whole integer interpolations can be implemented through simple flip-flops. It is only the fractions of a sample that are of concern. Thus, n in the above equation is always 0 making the coefficients of the interpolation filter:

$$h(k) = \text{sinc}(f_s\Delta n + k) \tag{5.2}$$

The coefficients in equation 5.2 can easily be stored in a lookup table for each value of k. This means that the number of lookup tables will equal the

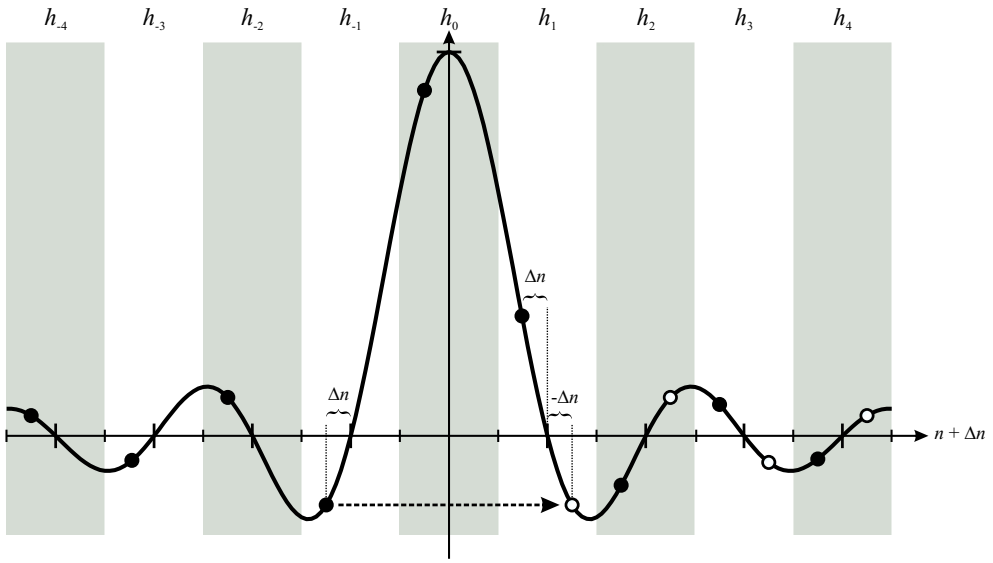


Figure 5.27: Interpolator FIR Coefficients from Sinc Function. The even nature of the sinc function allows for the use of half of the LUTs needed given that two values can be calculated from one LUT.

number of taps in the FIR. Using a lookup table for each coefficient also allows the use of windowing since each table can have values pre-multiplied by the windowing function. This simplifies the end logic. Figure 5.27 shows how each table, designated by h_n, would be arranged. The even nature of the sinc function allows a single table for $h_{\pm n}$ with the index for the positive n coefficients being based off Δn and the negative n coefficients being based off $-\Delta n$. Using the dual-ported Xilinx BRAM primitive, both values can be accessed simultaneously reducing the number of lookup tables, or BRAMs, by almost half.

As stated in section 3.4.2, at least 70 million values ($\approx 2^{26}$) are needed for each coefficient. The size of a single BRAM holds 1024 (2^{10}) different values. Linear interpolation can be used to approximate enough values in between those stored in the tables by using a second set of tables to store the slope at each point. Figure 5.28 shows how the slope can be used to calculate both $\pm\Delta n$. The error is unique depending on which direction is being interpolated. There are techniques to reduce this error by adjusting both the stored value and slope to minimize the

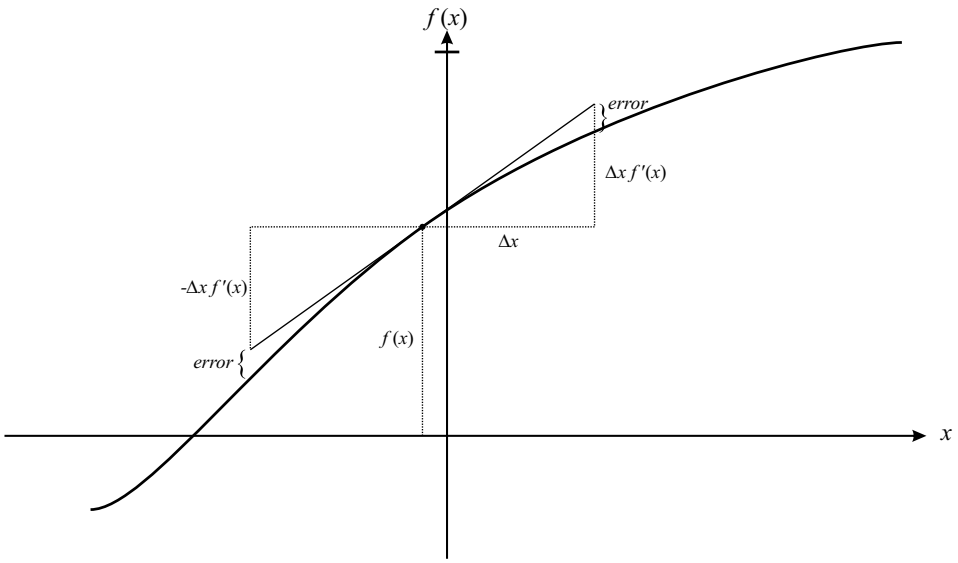


Figure 5.28: Interpolation FIR Coefficients Increasing Resolution using Linear Interpolator. Linear interpolation allows for both positive and negative values of Δx using a signed multiplier.

maximum error [6]. However, these techniques don't take into account this dual value approach, although adapting them could be quite simple. The resulting error, it turns out, is sufficiently small without any adjustments for the precision needed in this implementation.

The implementation is quite simple using the BRAMs. Figure 5.29 shows the design using very few primitives and minimal logic. The multipliers required are implemented with the 18-bit signed hardware multipliers inside the Virtex-II. An 18-bit signed number allows for 17-bits of precision in the positive direction. Used in conjunction with the 10-bits in the address of the BRAMs, this comes to $2^{10+17} = 2^{27}$ positions between samples that can be interpolated which exceeds the required 70 million positions. This simple design allows for a theoretical delay granularity of $\frac{1}{2^{27} \times 56MHz} \approx 1.33 \times 10^{-16}$ seconds along with the ability to dynamically change the coefficients of the filter with each clock cycle, thus changing the delay.

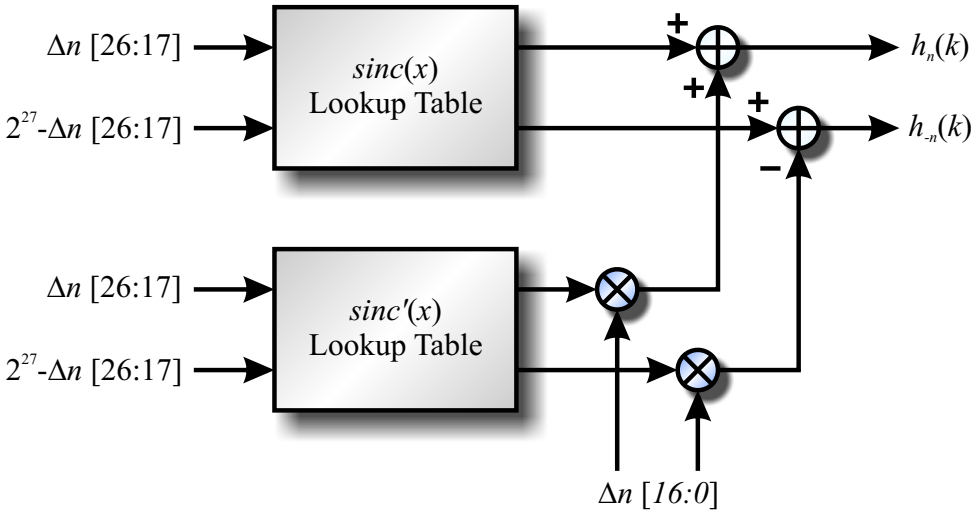


Figure 5.29: Block Diagram of Interpolator Coefficients Calculation using LUTs and Linear Interpolation.

Filter Design

With the coefficients calculated, the design of a finite impulse response (FIR) filter is needed. The basic design is based of a simple FIR digital filter as shown in figure 5.30. The number of taps, and subsequently the number of coefficients needed, is decided be finding the minimum number of taps needed to meet the accuracy required for a given frequency range. For simplicity, only a 3-tap filter is shown in figure 5.30.

The unique aspect of the entire interpolator FIR filter design is its ability to change dynamically. As mentioned, this is used to essentially speed up or slow down the playback of the signal stored in the digital delay to best mimic the Doppler Effect. In doing so, the interpolator progresses slightly faster or slower through the data. Since the interpolator only has a range of $\pm 1/2$ a sample, it will occasionally need to wrap around its range while simultaneously skipping or holding the next sample. This is the function of the throttle logic shown in figure 5.30. As can be seen, this logic can control the whole sample increments

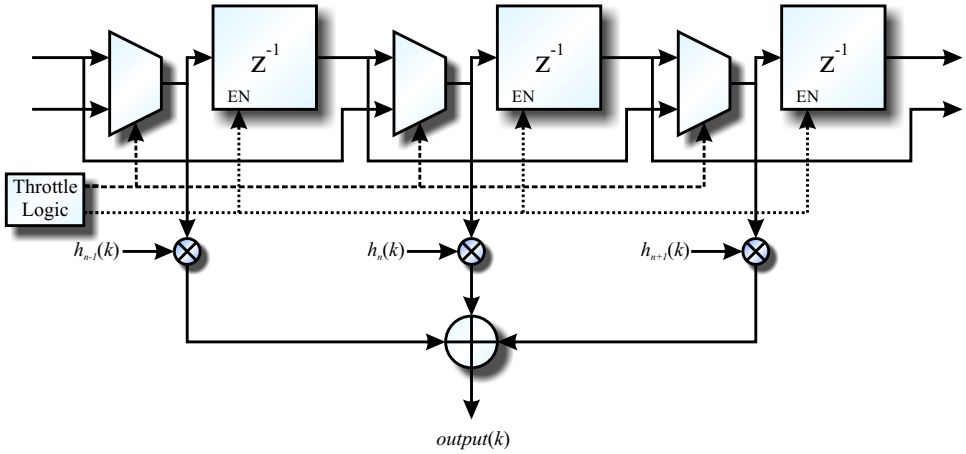


Figure 5.30: Interpolation FIR Block Diagram. Modified from a basic FIR design, this FIR has the ability to progress by 0, 1, or 2 samples per clock determined by the throttle logic component.

or decrements that are required for this particular FIR. It is this logic, too, that communicates with the attached memory controller to ensure continuity throughout operation.

Once the correct samples are multiplied with the correct coefficients, the result of each must be summed together. The design of this final adder takes some consideration when implementing it in hardware. A simple approach would be to simply create an adder for each tap essentially summing them serially. This implementation would result in a very slow final adder. Without increasing hardware, some of the additions could be done simultaneously. The most efficient use of parallel adders would pair the terms and add each pair simultaneously, then pair the results and add simultaneously, and continue until a single result was found. Although this reduces the computation time significantly, there are more efficient implementations.

Summing the terms of each tap in an FIR is similar to the summing of the partial product terms in a multiplier. There are many techniques that use a form of compressor, or commonly called a counter, to quickly and efficiently reduce

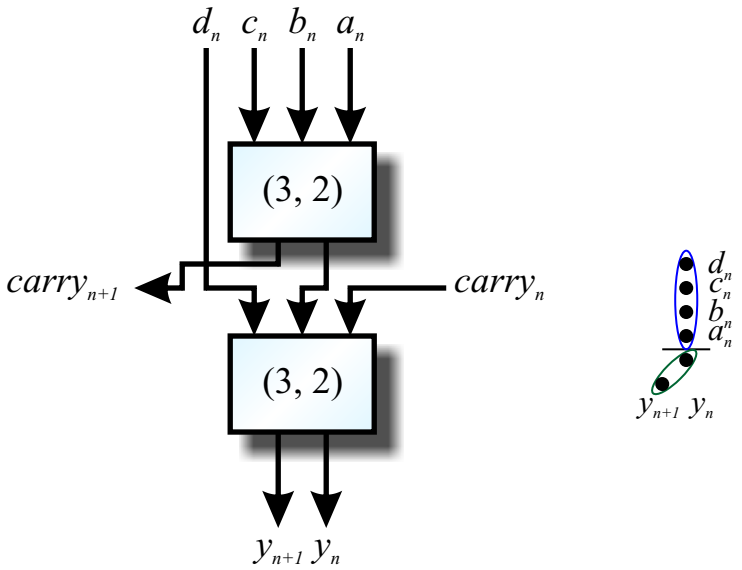


Figure 5.31: Basic Implementation of a 4:2 Counter using 3:2 Counters. The propagation of such a setup will be that of two 3:2 counters at most since the carry logic only propagates as far as the adjacent bit.

the total terms down to two which can be handled with a simple adder [7]. The most basic counter is a 3:2 counter which, logically, is identical to a single-bit full adder cell. This essentially adds the three input bits of equal order of magnitude and produces a two bit result.

Another common counter is the 4:2 shown in figure 5.31. This is designed with two 3:2 counters. The design has 5-bits being added together (the four inputs along with a single carry in bit) to produce a 3-bit result, one of which is a carry out. The carry bits are tied to the adjacent counters. The use of the 4:2 counter allows the number of resulting terms to be half that of the input terms while keeping the delay constant regardless of the width of the terms being added. This means that 16 terms can be reduced to two terms in three stages as shown in figure 5.32.

Given the use of counters and the variety of adder topologies, there are many different ways to implement the final adder needed for the FIR. A summary of

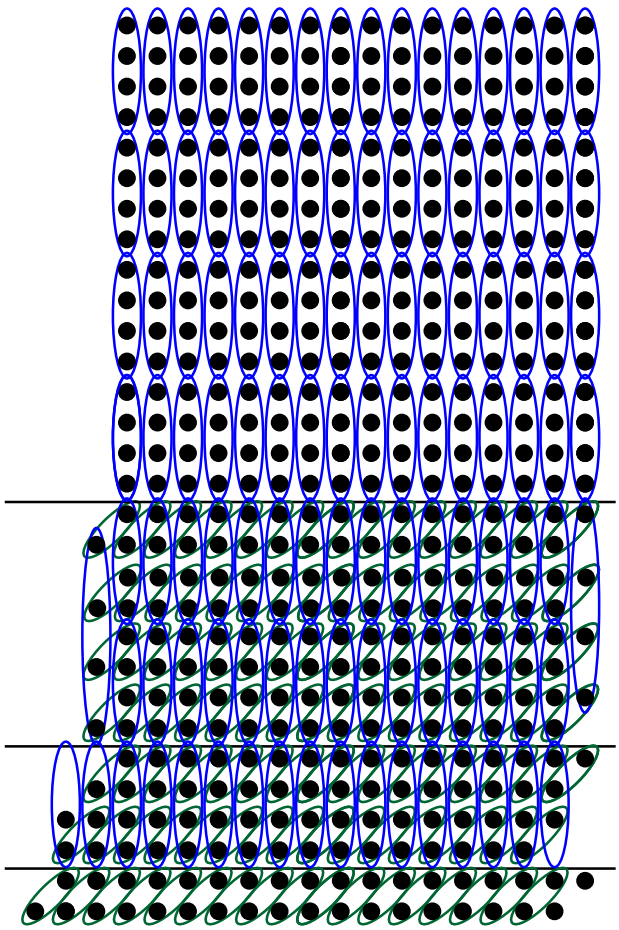


Figure 5.32: Design for a Parallel 16 Value Adder Using 4:2 Counters. The propagation delay will be the same through each stage, thus the total delay will be determined by the number of stages: $\log_2 M - 1$ where $M = 16$ in this instance.

these are shown in table 5.1. The size and speed listed are in reference to the number of FPGA primitives needed to implement the different designs. The primitive in the Xilinx FGPAs is a section of the internal slices used for the combinational logic called a lookup table (LUT). Both a single bit full-adder cell and most parallel prefix operators can be implemented in a single LUT. This defines the total number of LUTs in the design while the speed is estimated by the largest number of LUTs that any single input must go through before reaching an output.

There are a few assumptions that should be clarified when looking at the listed delay times in table 5.1. First, the delay time does not take into account the routing or fanout that will be required for each design. Although these have significant additions to the total delay, in most cases, they should be fairly proportional to the number of LUTs themselves. The next assumption is that the LUT is used exclusively in the full-adder cell. This is actually not the case for the Xilinx slices. Much of the carry logic can be implemented in a faster carry chain logic that bypasses the LUT. This greatly reduces the time for the ripple carry topologies and should be considered when reviewing the results listed.

Table 5.1 shows that the fastest approach would be to use the 4:2 counter reduction with a Kogge-Stone adder for the final two-term addition. The best use of space is with the 4:2 counter reduction with a simple ripple carry adder. There are two criteria used to determine the final adder design: to take as little space as possible and to meet the timing requirements. In other words, it must be able to operate at 56 MHz while minimizing the FPGA resources. Through experimentation, it was determined that the 4:2 counter reduction with the ripple carry adder could be implemented while not violating any of the timing constraints with the help of the fast carry-chain logic.

Adder Configuration	Number of LUTs $N = 32$	$M = 16$	Delay in LUTs $N = 32$	$M = 16$
Ripple Carry (Adders in Series)	$(M-1)N$	480	$(M-1)N$	480
Ripple Carry (Parallel Adders)	$(M-1)N$	480	$N \log_2 M$	160
Parallel Prefix Kogge-Stone in Parallel	$(M-1)(N \log_2 N + 1)$	2372	$(\log_2 N + 2) \log_2 M$	26
4:2 Counter with Ripple Carry	$N(\log_2 M - 1) + N$	160	$2(\log_2 M + 1) + N$	40
4:2 Counter with Kogge-Stone	$N(\log_2 M - 1) + (N \log_2 N + 1)$	316	$2(\log_2 M + 1) + \log_2 N + 2$	14

Table 5.1: Multiple Adder Configuration Comparison. The numeric values given are with an example of adding 16 32-bit numbers together.

Performance

The spectral response of a perfect interpolator should be flat–a simple all-pass filter with a linear phase response. As with all digital filters, an ideal response can never be achieved with a finite impulse response. With an interpolor, the limitations of an FIR implementation can best be seen in the worst case response of interpolating half way between two sample points ($\Delta n = 0.5$). This can be seen in the spectral response shown in figure 5.33. Responses are shown for Δn =0.1, 0.25, and 0.50. As Δn approaches 0.5, the worst-case response is shown.

The final implementation was with a 31-tap FIR. The coefficients of the filter are multiplied by Kaiser window [13] with $\alpha = 7.0$. Windowing helped achieve a flatter response in band while sacrificing more of the out of band performance in the higher frequencies.

Aside from having a flat response, the ideal interpolator will also have a perfectly linear phase response where the slope is proportional to the effective fractional delay. The phase error can thus be another means of evaluating the performance of the final implementation. This, too, is shown in figure 5.33. As with the magnitude response, the worst-case can be found when the fractional delay, Δn, is higher.

As can be expected with an interpolator, the performance at the lower frequencies is near ideal and progressively worsens as the frequency increases. This phenomenon can be explained with a simple intuitive understanding as to what happens at the two frequency extremes. At the lowest frequency, the individual sample values are of all the same value making any interpolation between them easily understood to be that same value. At the highest digital frequency, $F = 0.5$, every sample is the negative value of that preceding and following. Thus, any interpolation halfway between will always have a magnitude of zero. This pattern of performance degradation with frequency is seen in both the magnitude and phase performance.

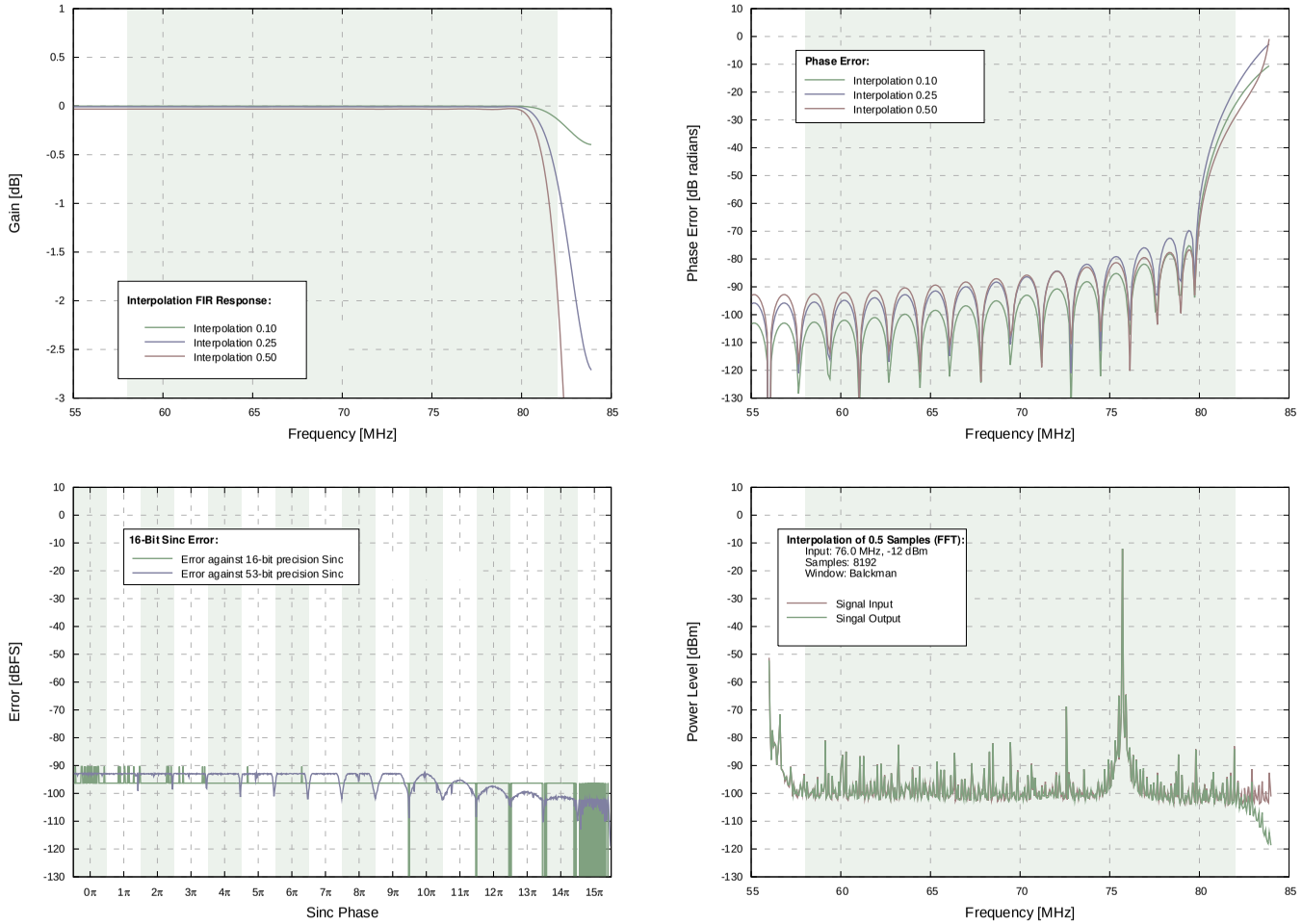


Figure 5.33: Performance Results of Sinc Function Generator.

The final performance of the overall filter is driven by the accuracy of the filter's coefficients. This is also shown in figure 5.33. An exhaustive search was performed on all possible calculations of the coefficients including the interpolated values. There are two main ways the error can be calculated: by comparing the output value with the actual desired value (in this case estimated by doing a double floating point calculation which has 53-bits of precision), or by comparing the output to the desired value rounded to the precision of the output value. Both are shown in the figure. One would expect the error of the first to be within ± 0.5 LSB of the desired value while the second to have an error of zero. Given the imperfect nature of linear interpolation, error is shown to have ± 1.5 LSB. When compared to the equivalent of a value of 1.0, the full-scale value, these errors only come to a maximum of roughly -90dB.

The overall performance of the interpolation filter can be shown using sinusoidal injection testing where the magnitude response can be seen as well as exposing any introduced noise. Using an analog input signal, an FFT was used to show the frequency response in figure 5.33. Given the other performance measures, it can be expected that there would be very little noise introduced even at the worst-case interpolation of $\Delta n = 0.5$. This is what is shown in the results of the FFT as well: negligible introduced noise due to the interpolation. It can be easily seen that at the higher frequencies the response tapers off as it does in the predicted response.

Aside from using the interpolator to add a fraction of a sample delay, the design allows for a dynamic change in the delay essentially enabling a scaling of the data rate. Given a granularity of 2^{27} between samples that the interpolator allows, a scaling factor of down to 1 ± 2^{-27}. As mentioned in section 3.4.3, this scaling factor is applied in the frequency domain as if the signal were down-converted to a center frequency of 14 MHz.

The performance of the scaling, in theory, should be better than that of the interpolated signal held at a static value of $\Delta n = 0.5$, the worst case response for the

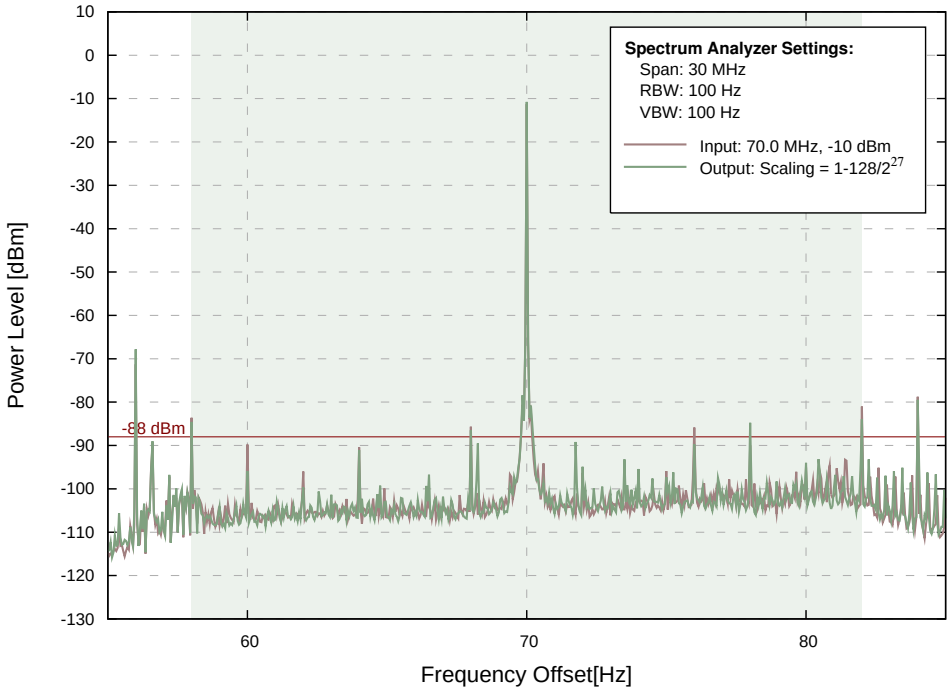


Figure 5.34: Spectrum Analyzer Output for Dynamic Interpolation.

interpolator. During dynamic, incremental changes to the interpolation location in order to achieve time and frequency scaling, the response moves predictably through the range of response performances. This slight periodic variation in performance could possibly add unwanted noise by itself. By injecting a sinusoidal signal and using a spectrum analyzer to record the output (the previous FFT's used do not have the resolution to confirm the accurate performance of dynamic interpolation), added noise can be measured along with the accuracy of the functionality. Figure 5.34 shows that no discernible noise is introduced due to the dynamic interpolation. Figure 5.35 shows the accuracy of the functionality. With an input of 70 MHz, the change in frequency should be $\Delta f = f - \alpha(f - 56\text{MHz})$ where α is the scaling factor. The two instances shown in figure 5.35 should produce $\Delta f \approx -1.04$ Hz and $\Delta f \approx -13.35$ Hz. These are the values measured on the spectrum analyzer's output.

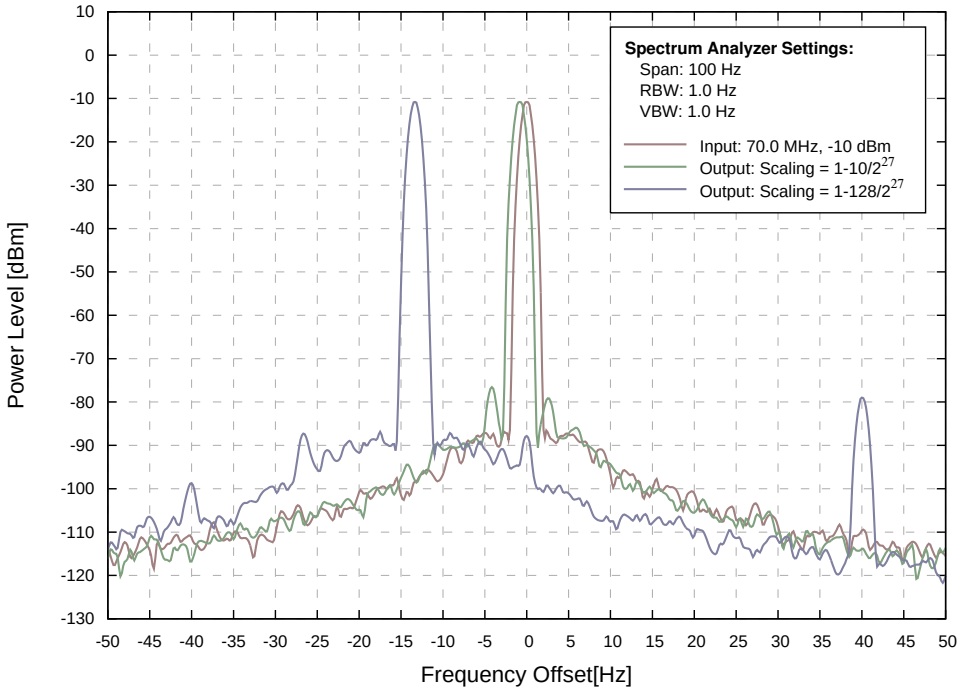


Figure 5.35: Detailed Spectrum Analyzer Output for Dynamic Interpolation.

5.2.3 Digital Single Side Band Modulation

Single side band modulation is necessary to complete the emulation of the Doppler effect as discussed in section 3.4.3. This static shift was shown to have to be able to do shifts down to 1.0 Hz in order to meet design requirements. With every modulation method presented, the resolution comes down to the granularity at which the sine and cosine function generators' frequency can be set.

Methods for Digital SSB Modulation

Presented in section 3.4.3 were three approaches to producing digital SSB modulation, the Hartley method, the balanced or matched Hartley method, and the Weaver method. It was determined that the balanced Hartley method has benefits over the standard Hartley Method. While the Weaver method should perform better than the Hartley method with an FIR implementation, its added complexity makes it less attractive. As a proof of concept, the Hartley method was originally choosen, while the balanced Harley method was implemented to verifiy any improved performance.

Both the standard and balanced Hartley methods can be implemented using virtually the same logic. Figure 5.36 shows the simplicity of the two implementations with simple substitutions regarding the two approaches. In order to show the validity of the proposed design, it would be best to first understand the digital Hilert transform.

An ideal Hilbert transform has the following impulse response and thus filter coefficients [6]:

$$h(n) = \frac{1 - \cos \pi n}{\pi n} \tag{5.3}$$

This will produce, as defined, a $-\text{sgn}(f)90^\circ$ phase shift. In order to produce an

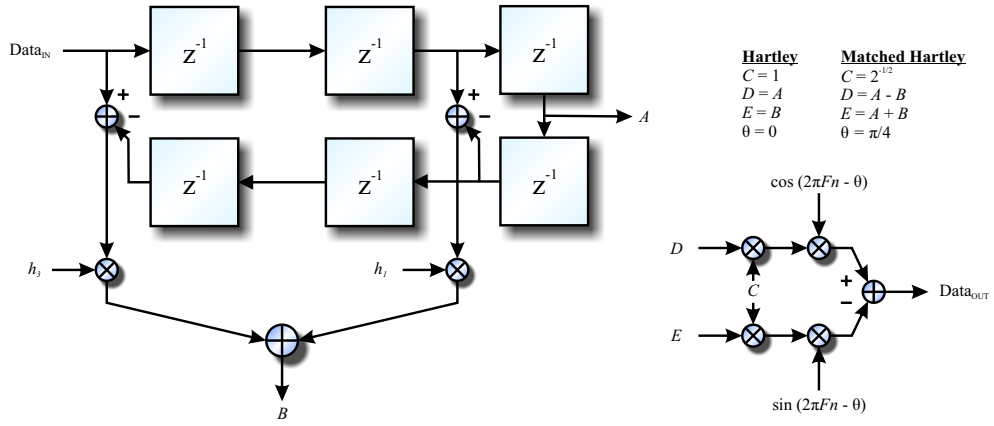


Figure 5.36: Block Diagram of Standard and Matched Hartley Methods using a Hilbert FIR Transformer. This block diagram shows how the same Hilbet FIR transformer can be used for both the standard and matched versions of the Hartley method for SSB. This example shows a 7-tap FIR for simplicity. The odd nature of the filter coefficients and the fact that all even coefficients are zero help to simplify the logic.

arbitrary phase shift, the following coefficients can be used [6]:

$$h(n) = \begin{cases} \cos\theta, & n = 0 \\ -\sin\theta \dfrac{1-\cos\pi n}{\pi n}, & |n| > 0 \end{cases} \tag{5.4}$$

Here, θ is the desired phase shift in the form of $\text{sgn}(f)\Theta$

The characteristics of these coefficients allow for many design optimizations. The most prominent is their odd nature. This allows for the use of simple subtracters, in the form of modified adders, to reduce the number of multipliers used as shown in figure 5.36. Aside from the center coefficient, $h(0)$, in the arbitrary Hilbert transform, every even coefficient has the value of zero. This allows for the implementation of a 109-tap FIR implementation using only 27 different coefficient magnitudes, greatly reducing the logic required.

Implementing both the standard and balanced or matched Hartley method for SSB can be selectable given the nature of the standard Hilbert transform

and the partial Hilbert transform evaluated at $\pm 45°$. The output of the matched delay and the output of the standard Hilbert transform utilized in the standard Hartley method for SSB shown in figure 3.13 can be scaled and summed according to equation 5.4 to produce the needed paths for the balanced method shown in figure 3.15.

The performance of the implemented Hilbert transform, with 109-taps and using a Kaiser window [13] with $\alpha = 8.0$, is excellent in the frequency range of interest. This can be seen in figure 5.37. A Hilbert transform should be as flat as possible in the band of interest. This implementation is shown to have ripple contained within 0.0015 dB. The phase result of the implementation is shown to be almost without error within the accuracy of a double floating point number. This can be seen in the Phase Error measurements which show errors below -300 dB radians for the entire band of interest.

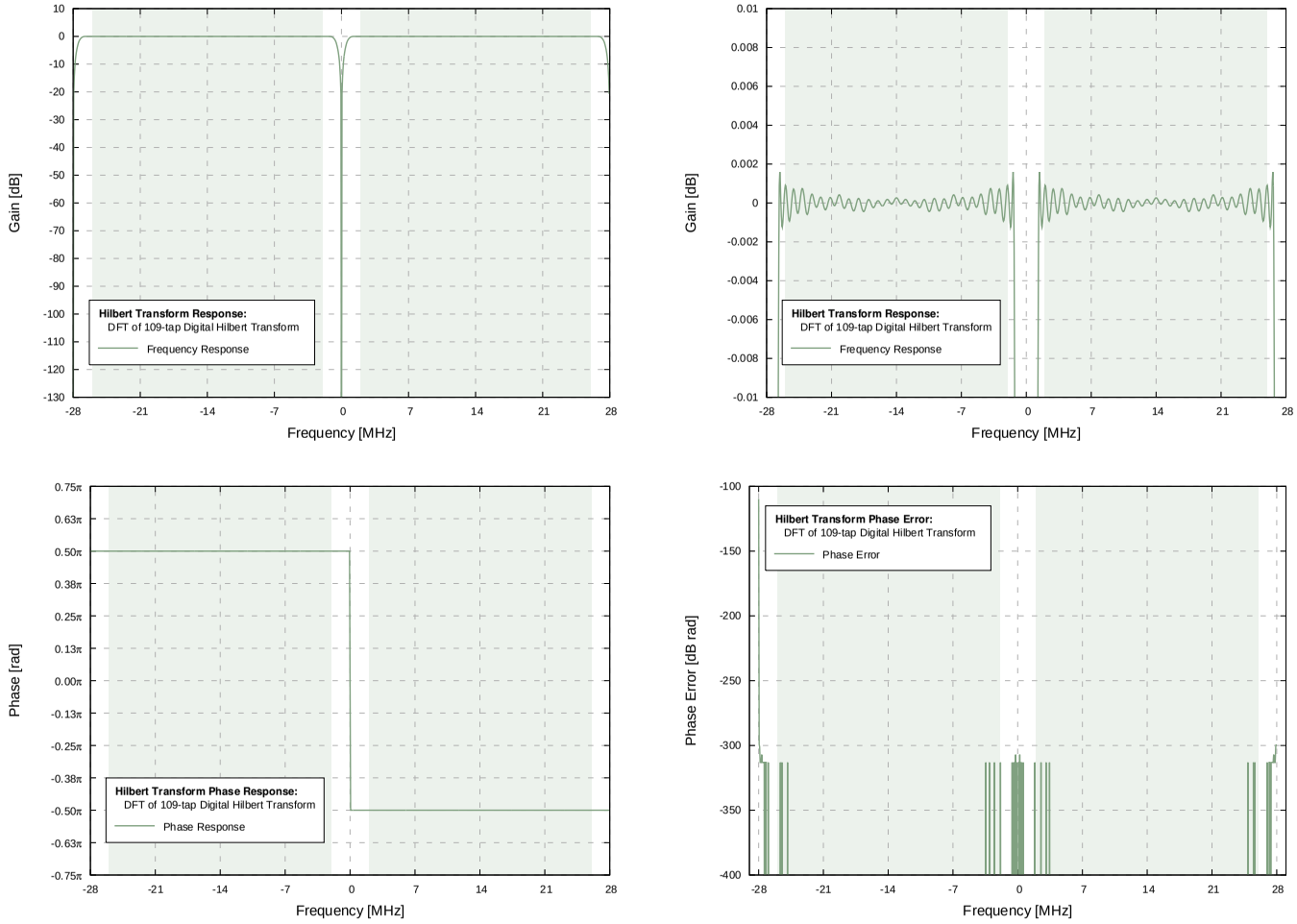


Figure 5.37: Performance Results of Hilbert Transform FIR.

Sine and Cosine Generation

As mentioned previously, the sine and cosine function generator needs to have the ability to set the frequency to within one Hertz. This creates a particular challenge when designing a system that can operate at 56 MHz. As with the sinc generator used in the interpolator, the sine and cosine functions can be implemented with a lookup table; however, this approach by itself would require 56 million entries to achieve the one Hertz granularity.

A decent approximation can be obtained using some sort of lookup table coupled with a form of interpolation. Simple interpolation could be done with the knowledge that $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\beta\sin\alpha$. If β is a relatively small value, $\cos\beta$ can be approximated by 1 and $\sin\beta$ can be approximated by β. To determine what β is, the number of final values in the sine generator needs to be known. The sine and cosine functions can be approximated in terms of α and β as:

$$\sin(\alpha+\beta) \approx \sin\alpha + \beta\cos\alpha \tag{5.5}$$

$$\cos(\alpha+\beta) \approx \cos\alpha - \beta\sin\alpha \tag{5.6}$$

For simplicity, much of the design is based on the available hardware within the Xilinx FPGA. 2^{10} entries can be stored in the dual ported BRAM. With a hardware multiplier for interpolation, a positive value of up to 2^{17} can be used. And, if the quarter symmetry of the sine function is taken advantage of, that is to say only a quarter of the sine wave needs to be calculated in order to produce a full sine wave, an additional 2^2 values can be used. This comes to a total of 2^{29} distinct points that can be calculated along the full sine period thus producing: $\sin 2\pi n 2^{-29}$ or $\sin 2^{-28}\pi n$ where n is between 0 and $2^{29}-1$. With a sample rate of 56 MHz, this comes to a resolution of $56\times10^6/2^{29} \approx 0.1043$ Hz, well within the requirements.

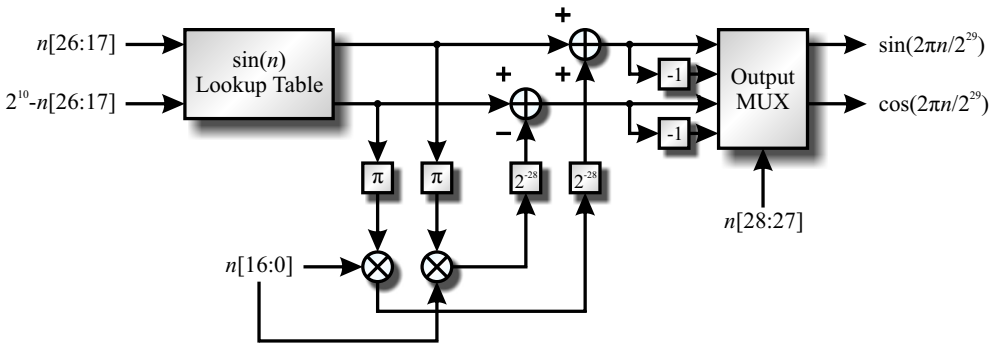


Figure 5.38: Block Diagram of Sine and Cosine Generator.

Looking at equations 5.5 and 5.6, both the sine and cosine of α are required to calculate either of the two functions. Thus, if both are obtained, it is relatively easy to simultaneously calculate both functions. In this instance, $\sin\alpha$ and $\cos\alpha$ represent the rough estimate stored in the BRAM. Given the dual-ported nature of the BRAM, both can be obtained simultaneously. Since only the first quadrant of the sine function is stored, the corresponding cosine value is simply equivalent to addressing from the end of the lookup table stored in the BRAM and moving towards the front. This is shown in figure 5.38. The value for β as described above is simply $2^{-28}\pi n$ where n is a value between 0 and $2^{17}-1$ representing the distance of interpolation between two points in the lookup table. Both $\beta\sin\alpha$ and $\beta\cos\alpha$ are implemented in the block diagram in figure 5.38 using a hardware multiplier along with a bit shift and hardwired π multiplier.

The hardwired π multiplier is accomplished using Booth encoding[8]. Booth encoding makes use of both adders and subtracters during the summation of the final product terms in a multiplier thus reducing the overall logic necessary. Used along with 4:2 and 3:2 counters, very low latency and space efficient multipliers can be designed. When multiplying by a constant, the encoding can be done before hand with a simple VHDL routine or by hand. Much of the FIR constants were implemented this way throughout the design to free up the built in hardware multipliers in the FPGA for cases when both the multiplier and multiplicand are

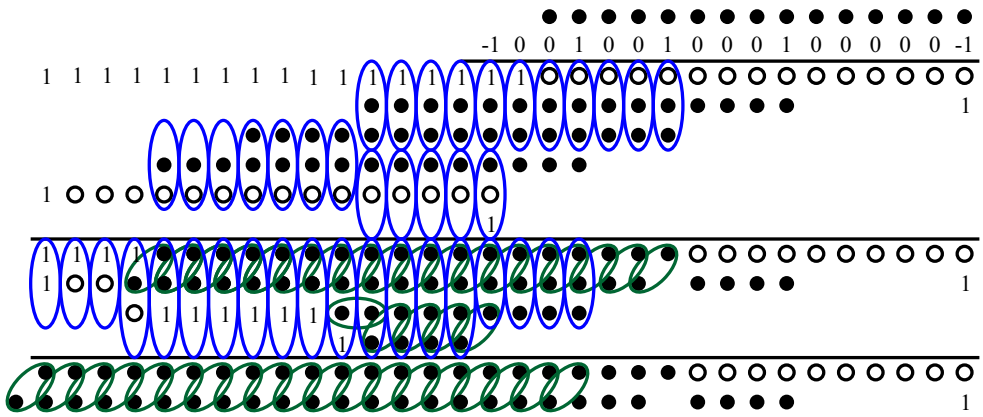


Figure 5.39: Implementation of a π Multiplier using 3:2 and 4:2 Counters and Booth Encoding.

variable. Figure 5.39 shows how the π multiplier was implemented in hardware. Subtractions were handled by simply inverting the bits and adding a '1'. With this implementation, the maximum delay is 3 LUTs before the final adder at the bottom making for a very fast multiplier.

Performance

The final performance of the two digital single side band modulation techniques implemented in the design are both influenced by the performance of the Hilbert transform and the performance of the sine and cosine components. The Hilbert transform performance is summed up in figure 5.37.

The sine and cosine generator is evaluated much in the same way as the sinc generator: by comparing the output values with the expected values. Figure 5.40 shows the maximum error along the entire period of the sine and cosine functions after an exhaustive search was done through every interpolated point. The error is determined against the double floating point precision (53-bit) calculated value. It is displayed in dBFS which is the ratio of the the error to the full-scale value, the equivalent of 1.0 from the sine and cosine generator. As can be seen in the

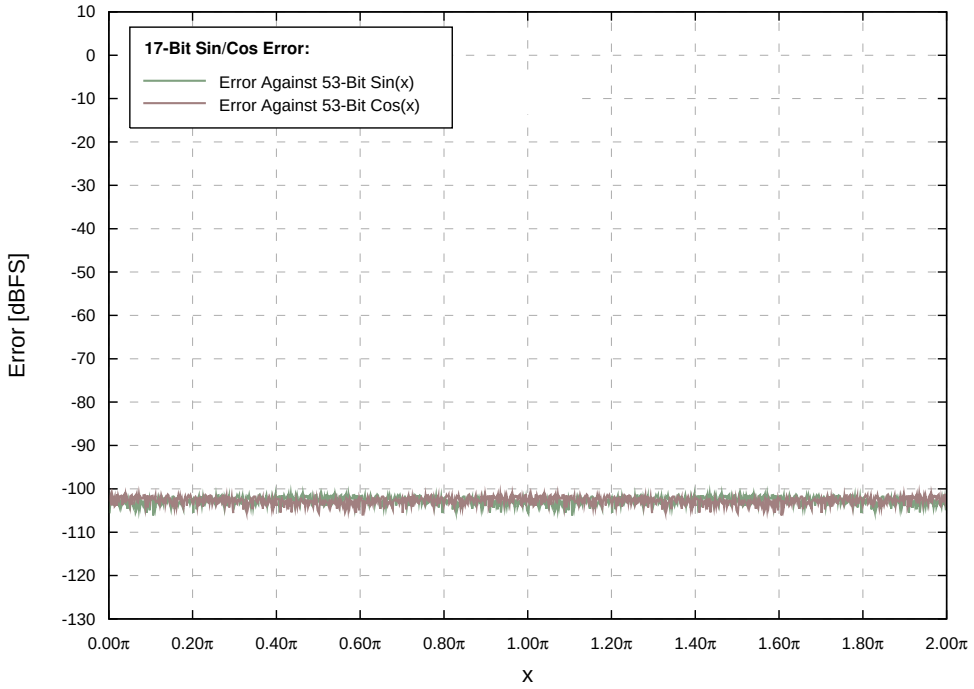


Figure 5.40: Sine and Cosine Error Analysis. The result of the sin/cos generator is an 18-bit signed number (17-bits of absolute value precision). An exhaustive search algorithm is used that searches for the worst error of the final output compared to a double precision float value (53-bits of absolute value precision).

figure, the error never exceeds -100 dBFS.

To determine what improvement the balanced Hartley method had over the standard Hartley method, both were used with the same sinusoidal input and desired frequency shift. The results are shown in figure 5.41. Here two input frequencies were chosen: one near the center and one near the edge of the band of interest. The incredibly flat response of the 109-tap FIR Hilbert transform resulted in next to no noticeable improvement using the balanced method. Both had near identical performance results with practically no measurable noise introduced due to the use of the SSB modulator.

The final output was measured again using a spectrum analyzer on the output as opposed to the FFT as was used in figure 5.41. Figure 5.42 shows the result

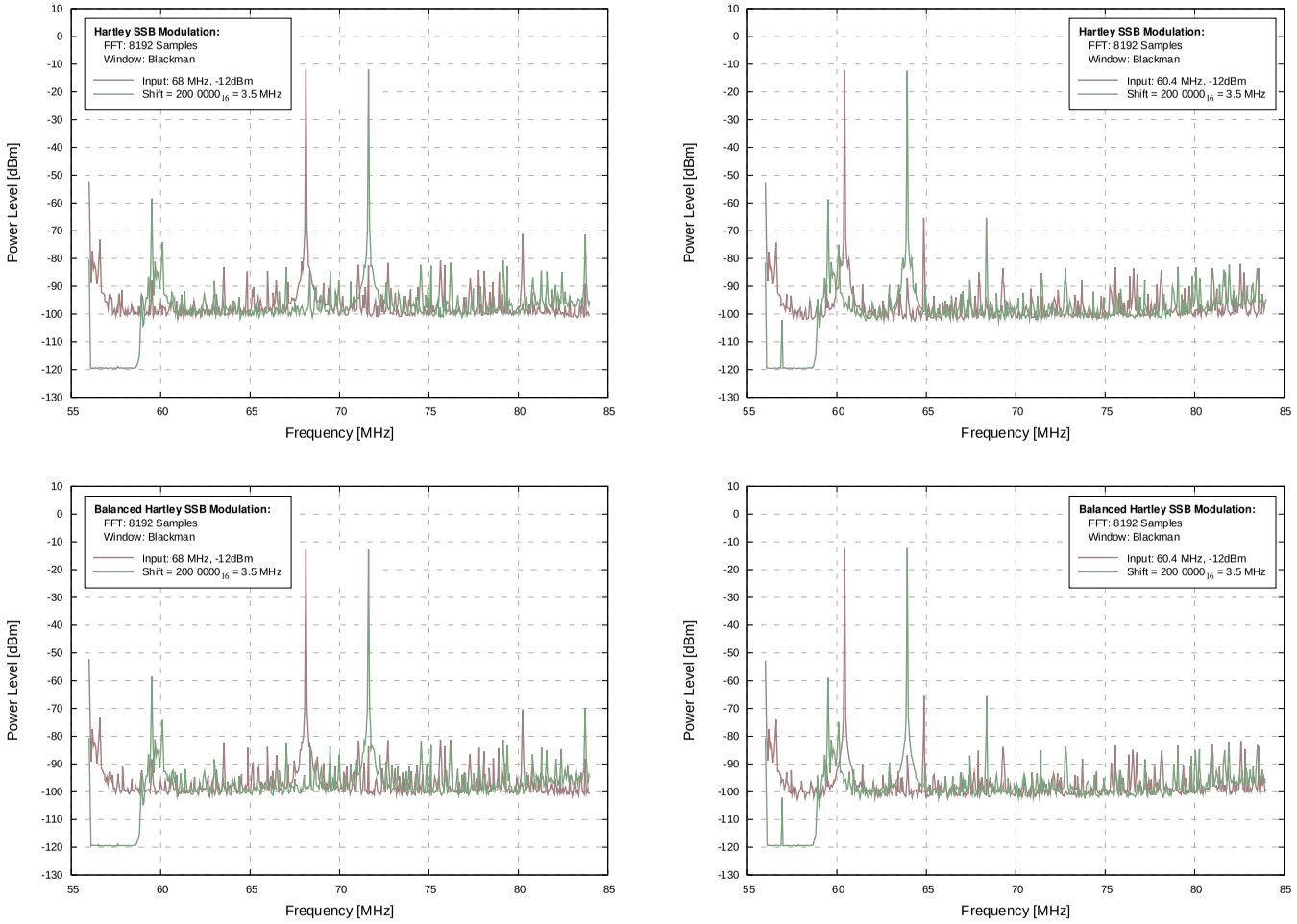


Figure 5.41: Performance Results of Sinc Function Generator.

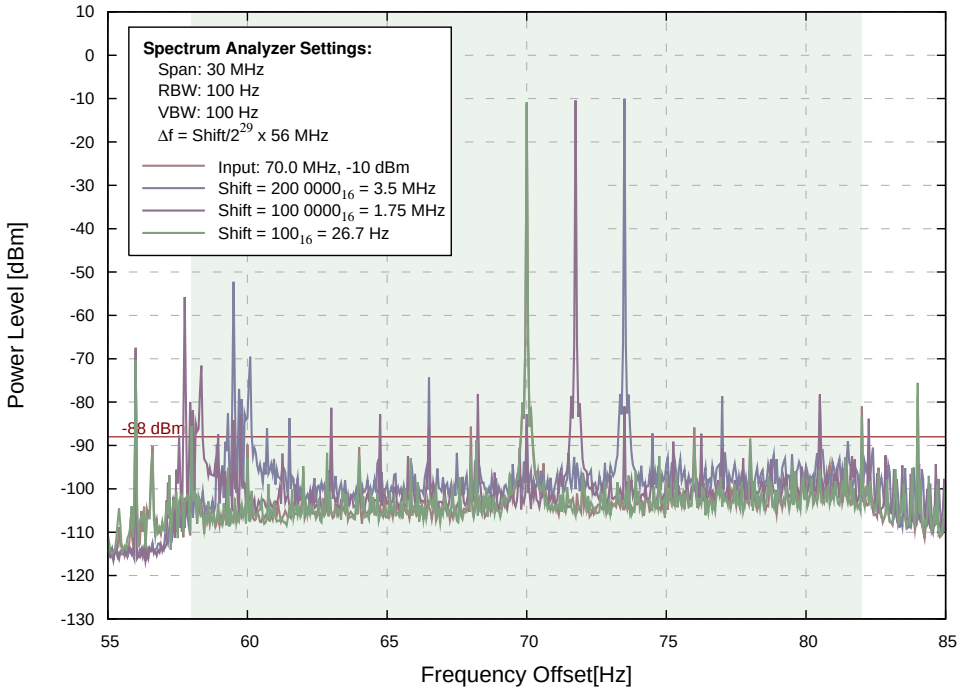


Figure 5.42: Spectral Plot of SSB Modulation.

of this test. Here, the noise floor is slightly raised with the increase in Δf. It should be noted however, that no noticeable increase in noise can be seen for small shifts below 1.0 MHz which is were the system will be operating for all foreseeable scenarios.

As to the precision in which the system can shift the frequencies, the input was compared to the output on an oscilloscope with a shift of 0.104 Hz. When triggered on the rising slope of the input signal, the scope showed a slowly moving output signal that would line up with the input signal once every 9.6 seconds confirming the small shift. To show a small shift statically, a spectrum analyzer was used as shown in figure 5.43. Here, a shift of 26.7 Hz can be seen. It should be noted that there is definite spurious noise introduced here which is not seen in figure 5.42. This most likely means that this noise lies close to the input frequency. Given the spurs relatively low power level and relatively small band in which they are introduced, they are most likely not significantly detrimental

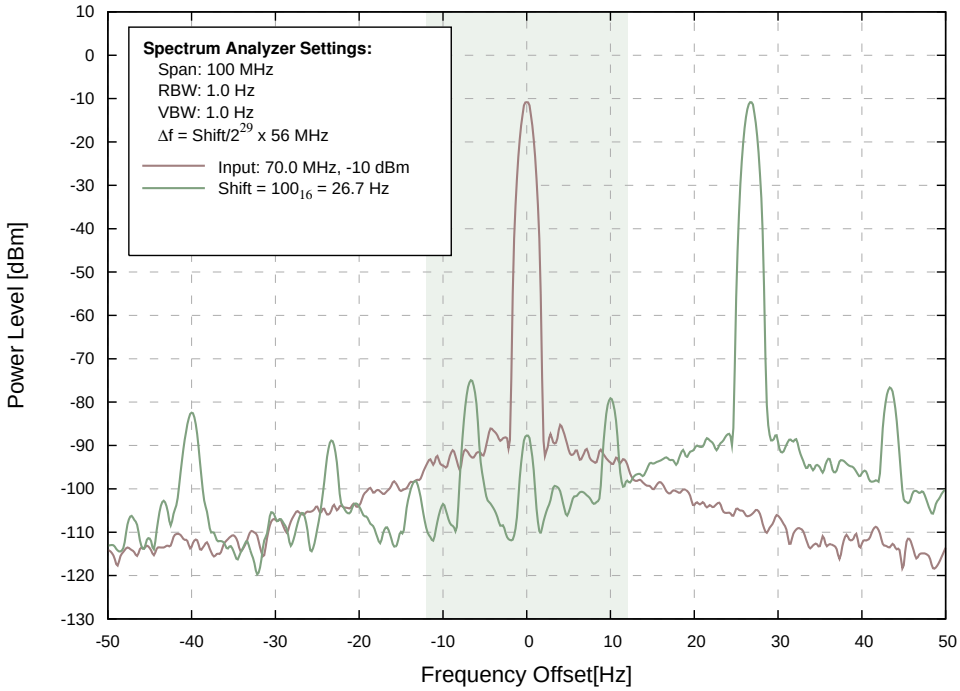


Figure 5.43: Detailed Spectral Plot of SSB Modulation.

to the overall performance of the SSB modulator.

5.2.4 White Gaussian Noise Generator

The Additive White Gaussian Noise (AWGN) component was implemented using a drop-in module from Xilinx. The module's probability density function (PDF) is rated as 0.2 percent Gaussian for $|x| < 4.8\sigma$ with a flat response. The algorithm used for generating the noise is based on the Box-Muller algorithm and the central limit theorem [9] [1]. The periodicity of the generated noise is reported to be 2^{190} samples. At a sample rate of 56 MHz, this comes to 2.8×10^{49} seconds. Given the current age of the universe at approximately 4.3×10^{17} seconds, one can safely assume this will never repeat without resetting the system. Based on the design discussed in [9], a periodicity of only 2^{60} is obtained. Still, this comes to approximately 653 years before the sequence repeats.

Desired		Best Fit Values			
σ	A	σ	A	Δx	$\sum Error^2$
1024.0	0.000390	1020.93	0.000391	-2.229	0.003885
512.0	0.000779	510.00	0.000782	-5.806	0.007773
256.0	0.001558	255.63	0.001561	-7.055	0.015530
128.0	0.003117	127.82	0.003122	-7.559	0.031054
64.0	0.006233	64.12	0.006222	-7.743	0.061906

Table 5.2: Best Fit Gaussian Functions for AWGN Data.

To verify the performance of the AWGN generator, the PDF and the FFT of various settings were recorded and analyzed. The measured PDFs are shown in figure 5.44. Each function was fit to a Gaussian curve as shown figure 5.45. A PDF with a Gaussian distribution should fit the expression:

$$\text{PDF}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.7}$$

For the purposes of this discussion, $1/\sqrt{2\pi\sigma^2}$ is substituted with A.

The AWGN generator's performance is evaluated by how close the desired standard deviation, σ, fits the measured value. Also, how well the PDF fits a Gaussian distribution is shown in the total error squared of the measured function compared to the fitted curve. The results are shown in table 5.2. There is less than a 0.35 percent error in the desired and standard deviation and very small total error. There is some anomalous shift in the curve as the standard deviation is lowered. This by itself is no cause for alarm since a shift will result in essentially a small DC offset added to the signal which will be eliminated in the analog output.

Aside from being Gaussian, the output needs to be white as well. This can be evaluated by the flatness of the frequency response. For this measurement, the FFT was taken directly from the output of the AWGN generator. Averaging was performed (100 x) to reduce the variance in the spectral plot. The results, shown in figure 5.46, depict very flat responses for the various noise "intensities"

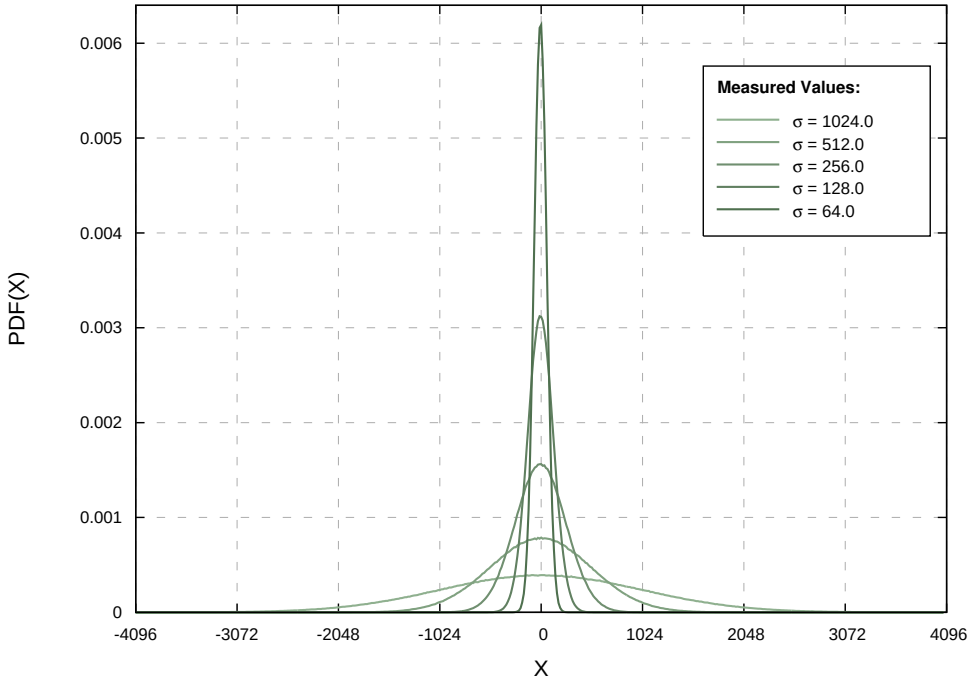


Figure 5.44: Additive White Gaussian Noise Probability Distribution Function Measurements.

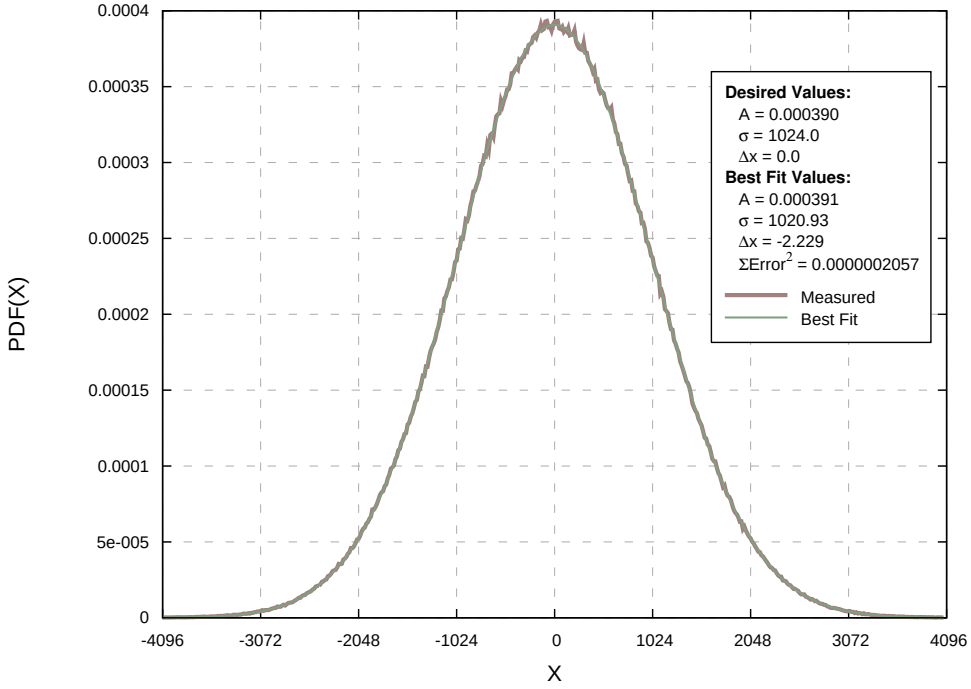


Figure 5.45: Fitted AWGN Probability Distribution Function.

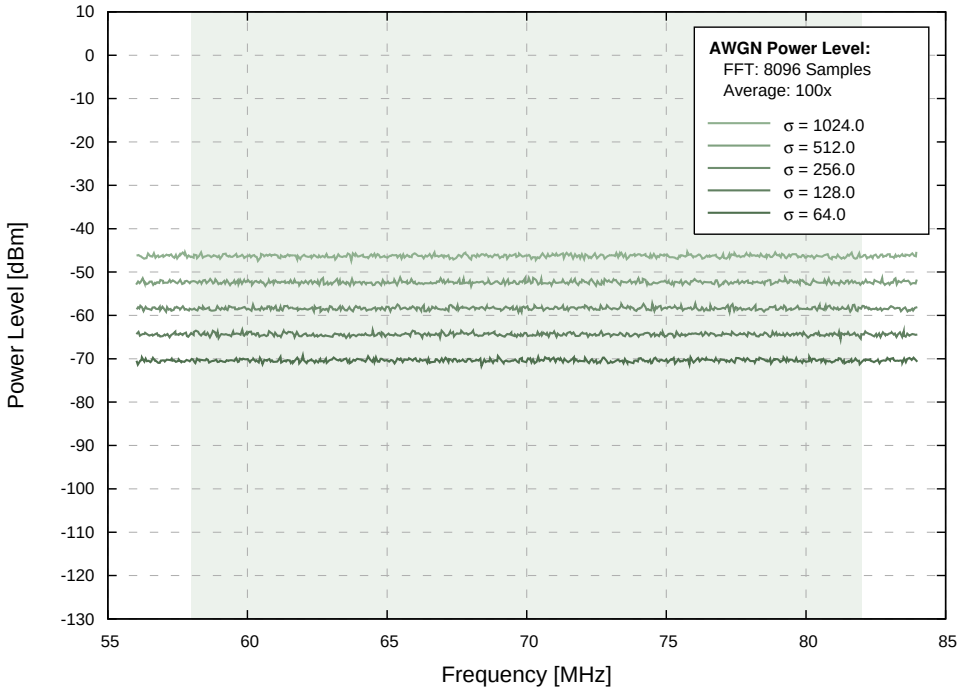


Figure 5.46: Additive White Gaussian Noise Spectral Plot.

measured. The maximum ripple being roughly ±2 dBm which can be attributed to natural variance given the sample size despite the averaging.

5.2.5 Digital Output Filter and Upsampler

As discussed in section 3.4.6, the frequency response of the channel emulator needs to be as flat as possible. The desired specifications for assuring flatness are ±1.0 dB maximum ripple across the band of interest. Much of the deviation from the flat response is due to the imperfections of the analog components. As discussed, these imperfections can be compensated for by measuring the natural frequency response of the system and designing a FIR filter with an inverted response.

It is this approach that was used to design the output filter. Figure 5.47 shows the natural response for the band of interest measured with a spectrum analyzer.

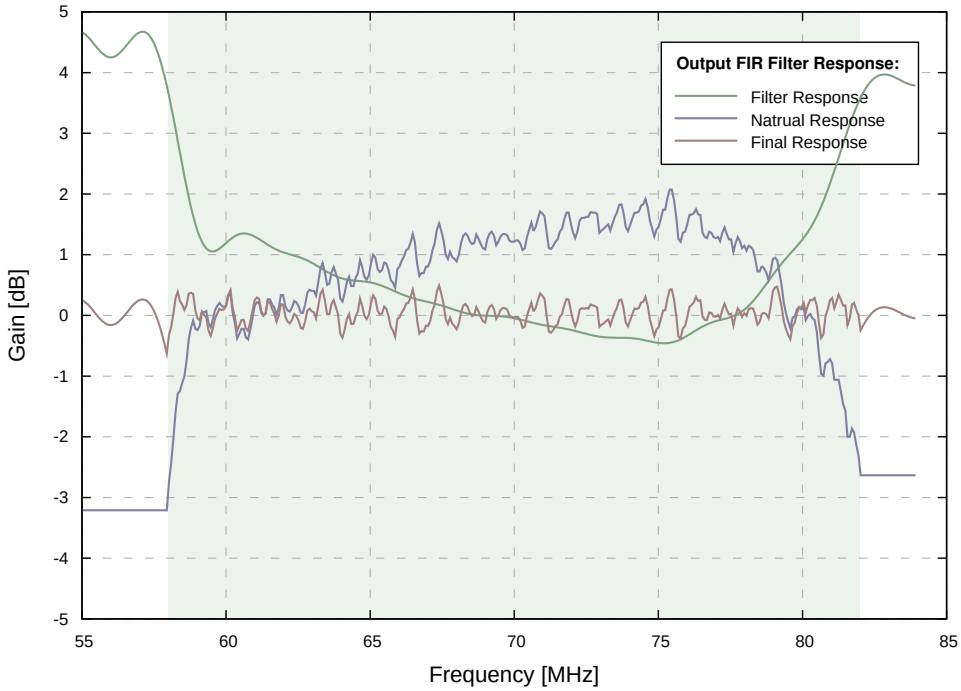


Figure 5.47: Theoretical Frequency Response of Output Filter.

The inverse of this response was processed through an inverse discrete Fourier transform (IDFT) to determine the coefficients for the FIR filter. Initially, this leads to an FIR filter with the same number of coefficients as the sample size of the measured response. Clipping the number of coefficients to a practical size (47-taps) effectively smooths the response of the filter allowing the higher frequency variance of the response to pass. This variance is below the specification of ± 1.0 dB. Figure 5.47 shows the response for the filter and the predicted output as well as the natural response of the channel emulator.

The implementation of the filter proved to be fairly straight forward in design with a final response that is fairly close to the theoretical prediction. The design, as mentioned previously, is based of a 47-tap filter. The coefficients were purposefully designed to be symmetrical to reduce the number of multipliers needed by a half as well as produce a near linear phase response. The multipliers were created using the constant coefficient multiplier design with Booth encoding as

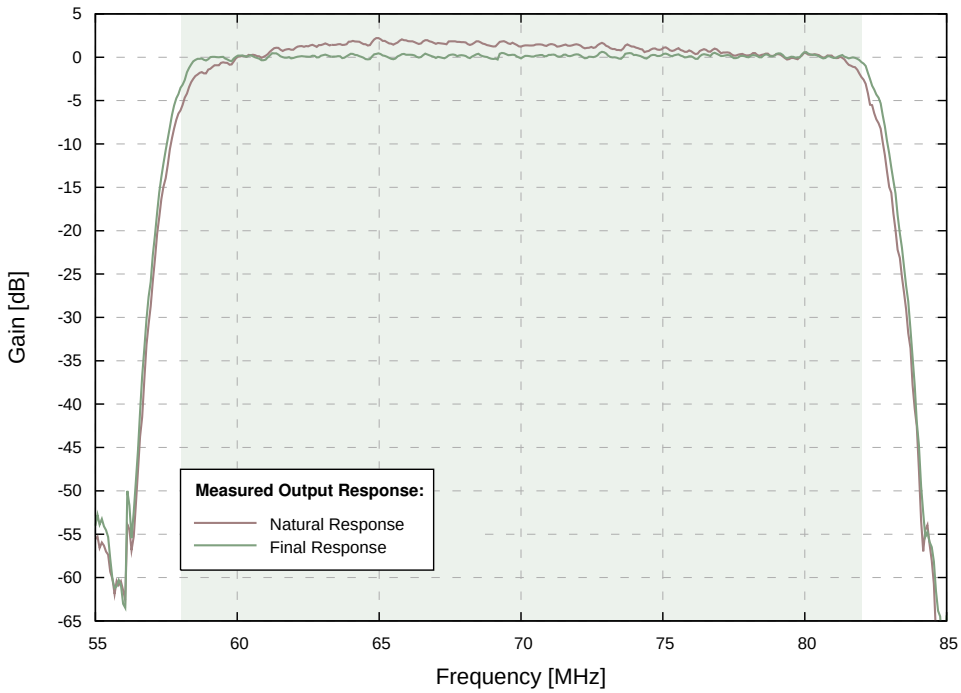


Figure 5.48: Measured Frequency Response of Output Filter.

was used in the π multiplier in section 5.2.3. The final response of the system is shown in figure 5.48. Here, the magnitude response in the band of interest holds to the desired specification of ± 1.0 dB. The out of band response is attenuated due to the bandpass filters at both the analog front and back ends.

The final upsampling phase in the design was implemented using dual data rate flip-flops at the output to double the sampling frequency. As mentioned, zeros are inserted in between each sample then the entire stream is passed through a high pass filter. The implementation of this can be seen in figure 5.49. Due to the fact that every other sample in both the data stream and the filter coefficients is zero, the logic is greatly reduced allowing for a much larger filter. The final implementation used a 119-tap filter which had the frequency response shown in figure 5.50.

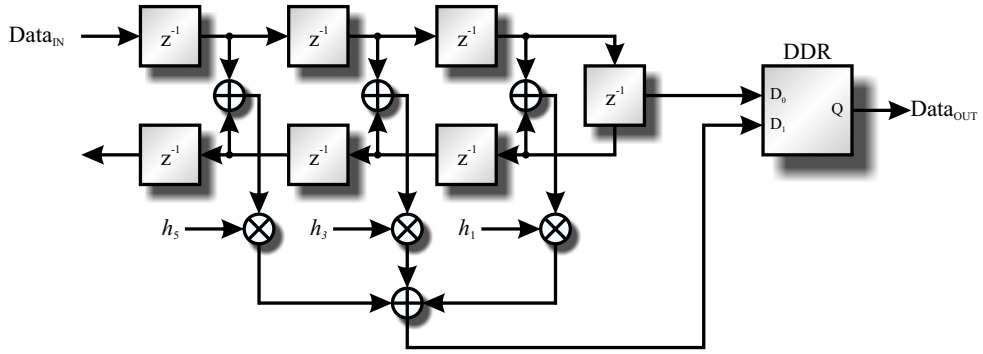


Figure 5.49: Block Diagram of Upsampler with Highpass Filter. The 119-Tap FIR HPF can be implemented with 31 multiplications taking advantage of the even symmetry and the fact that every other coefficient has a value of zero.

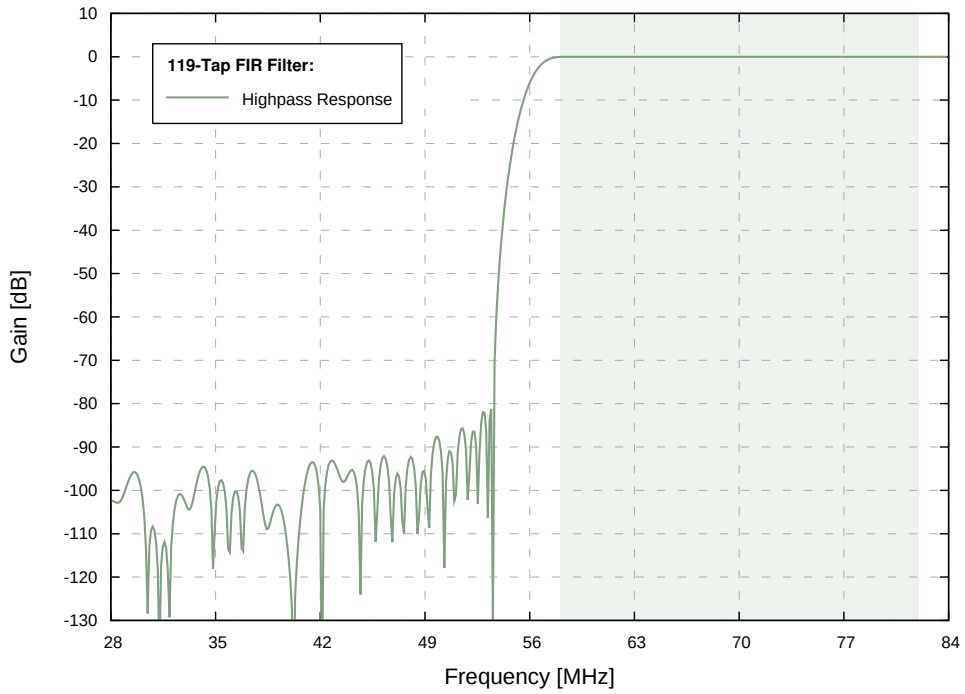


Figure 5.50: Frequency Response Highpass Filter.

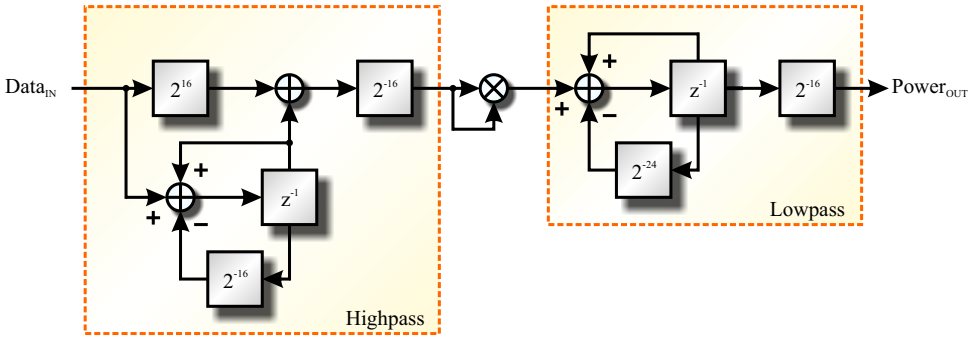


Figure 5.51: Block Diagram of the Power Measurement Component. Using simple 1st order filters allows for a simple implementation requiring minimal logic.

5.2.6 Power Measurement

In order to accuratly produce the affects of the automatic gain control (AGC) as discussed in section 3.4.5, both the power input and the power output of the channel emulator need to be known. Per the design requirements, the total power output should be held at a nominal -12 dBm. Since the signal to noise ratio (SNR) is a user defined variable, the introduced power of the additive white Gaussian noise (AWGN) generator is variable as well. Knowing the input power, or the signal power, and the output power allows for on-the-fly calculations of the input signal attenuation and the power level of the AWGN generator in order to meet the desired nominal output power and the desired SNR.

The power measurements need to be averaged over a certain time period. Instantaneous power cannot be held constant for any signal other than one of 0 Hz. The response time of the power measurements can be relatively slow given that most input signal's power will be fairly constant (≈-12 dBm) and SNR settings change very slowly under normal operating conditions for most space communications. The phase response of the power measurements can also be ignored for the most part because of the slower required response time.

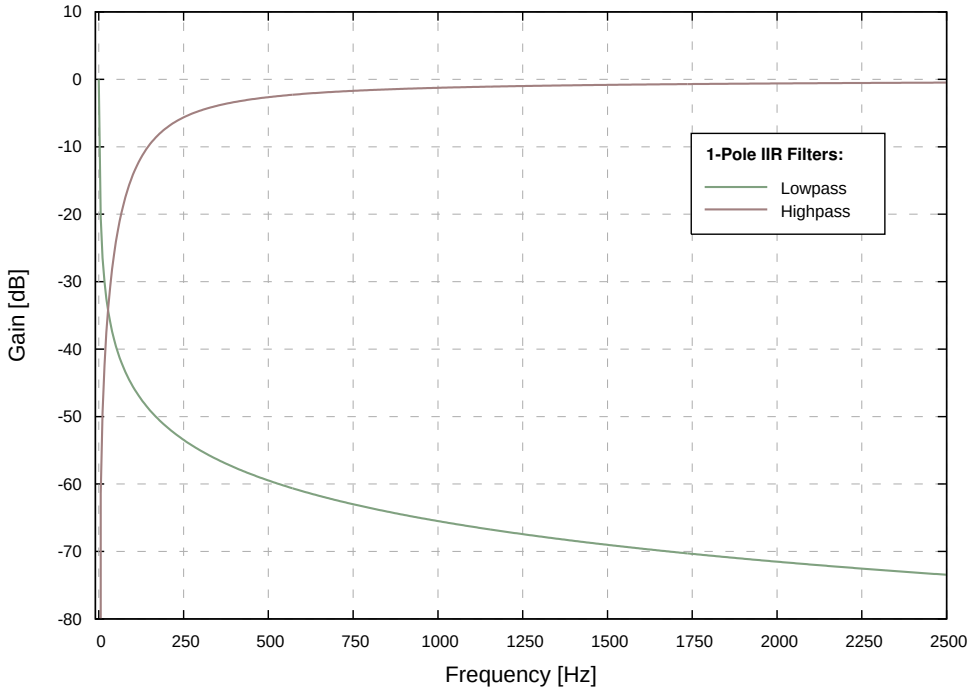


Figure 5.52: Power Measurement Component 1-Pole IIR Filter Responses.

Given these looser requirements and the desire to minimize internal FPGA component utilization, simple first-order IIR filters can be used. The block diagram for the design used is shown in figure 5.51. This diagram shows two major parts: a highpass filter and a lowpass filter. The highpass filter is necessary to eliminate any DC components that might be introduced through aliasing or otherwise. The lowpass filter is used as the "average" function. High frequency changes are attenuated from the output power.

There are two major metrics that can be used to characterize the performance of the power measurement: the frequency response and the response time, which are directly related. The frequency response of the highpass filter needs to be as flat as possible in the band of interest, while the lowpass filter needs to generally attenuate the higher frequencies. The results of the two can be seen in figure 5.52. This figure only shows up to 2500 Hz in the digital domain which translates to 2500 Hz above 56 MHz. This is still outside the band of interest, and both

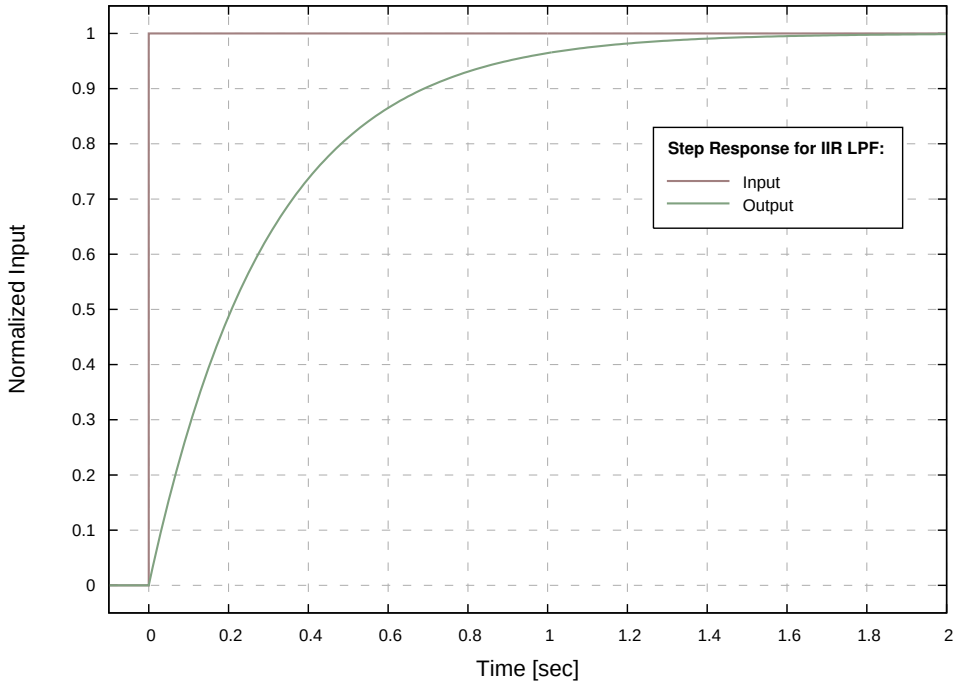


Figure 5.53: Step Response for the LPF of the Power Measurement Component.

responses are meeting their respective criteria.

The response time of the lowpass filter is slower than that of the highpass making it the time of interest. The only real criterion for this metric is that the system must be fast enough to be "usable". This means that anyone changing parameters manually must see results fast enough to feel the unit is working appropriately. This is a very general value that isn't explicitly defined. For this design the response curve is shown in figure 5.53. To reach 90% of the final value from a step input, the response time is approximately 0.7 sec. This seems adequate for usability.

The power measurement component was used to make power readings of the additive white Gaussian noise generator. For comparison, the results of are shown in figure 5.54 along with the results without the initial highpass filter, the results of using a spectrum analyzer to measure the analog output of the channel emu-

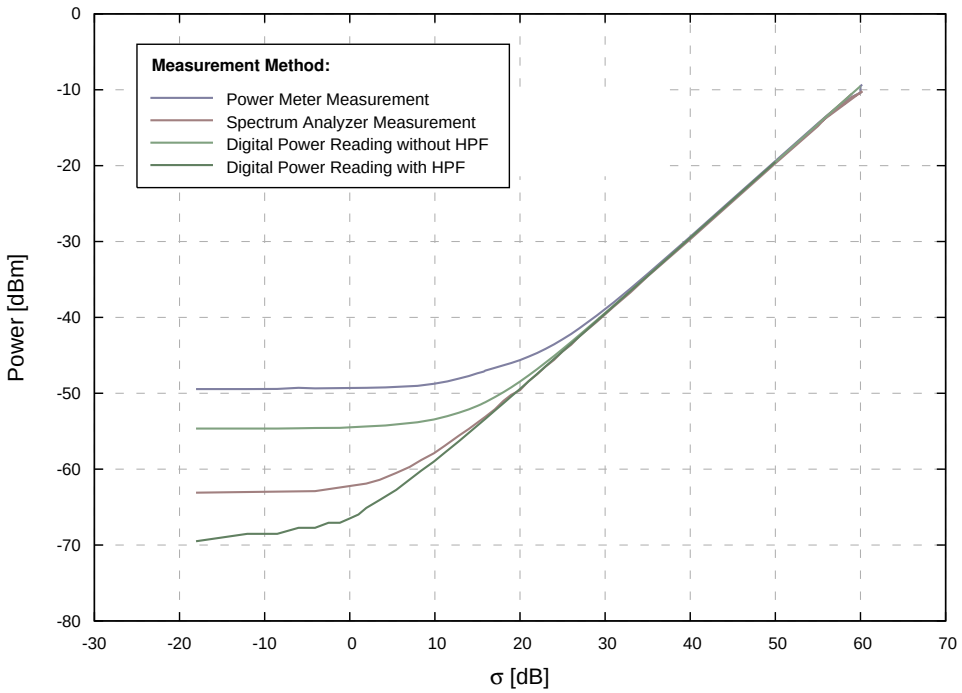


Figure 5.54: The Power Output of the AWGN Generator using Various Methods.

lator, and the results of using a power meter for the same analog measurements. When the standard deviation of the output of the white Gaussian noise generator, σ, is 30 dB and 50 dB, all measurements correspond relatively closely. Above that, the analog signals soon reach the compression points of the amplifiers in the analog output stage. Below 30 dB, the measurements asymptotically reach a minimum power level. This level is different for each method for various reasons. The power meter includes any out of band power that might be in the signal giving it the highest minimum power. Without the highpass filter, aliased DC noise is included in the minimum power of the digital power measurement component. The spectrum analyzer is limited by any analog noise inherent in the system. Finally, the digital power measurement component is limited by the accuracy of the AWGN generator along with any noise inherent in quantization.

Chapter 6

RF Channel Emulator System

In the previous chapter, each individual component was described and evaluated in detail. This chapter discusses the assembly of primarily the digital components, although since there was much change to the clock network, this is discussed as well. Finally, how the individual components communicate with the user interface is discussed explaining the respective contributions to achieving the overall goal of emulating a communication scenario.

6.1 System Assembly

As with much of this project, the system assembly is comprised of two components: the analog system and the digital system. The analog portion contains much of the physical hardware. This assembly is described in detail in the work done by Woolrich [22]. Deviations from Woolrich's work regarding the analog portion are discussed here. In addition, the digital components are discussed.

6.1.1 Clock Network

The largest change from the original work is in the clock network. The primary reason for the changes on a whole is to reduce jitter and spurious noise the clocks

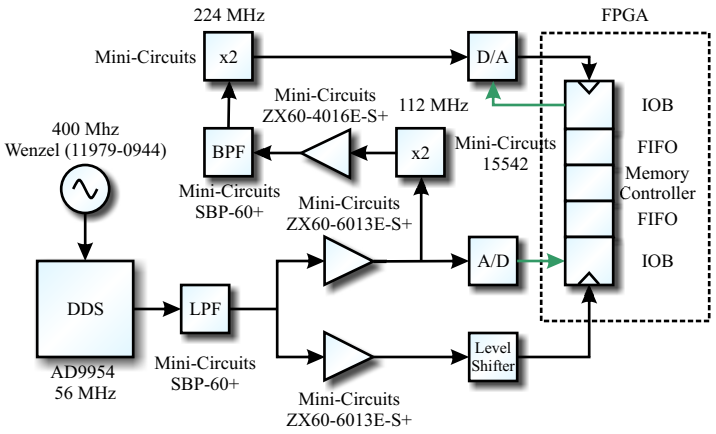


Figure 6.1: Block Diagram of the Clock Network.

were introducing into the system. This is discussed in great detail in section 3.5, section 3.6, section 5.1.4, and section 7.1.3.

The clock network was designed to ensure the proper frequencies were available for each hardware component along with the proper phase and voltage levels. A block diagram of this is shown in figure 6.1. The 56 MHz sample clock is created using the DDS run by the Wenzel 400 MHz oscillator. This is then split and buffered to drive the A/D and the input clock for the incoming digital data. These two paths were wired to have nearly identical path lengths to ensure proper phase between the data and clock at the input of the FPGA.

The clock is split at the input of the A/D and routed through the network for the D/A clock. First, the 56 MHz is doubled to 112 MHz. This is a lossy process and requires an amplifier to increase the signal strength sufficiently for the next doubler. A highpass filter is used to minimize spurious noise as discussed in section 5.1.5. The frequency is doubled again to 224 MHz to operate the D/A at 112 MHz input and 224 MHz output. Thus, the clock requirements of section 5.1.5 and section 5.1.5 are met.

6.1.2 Digital Components

Assembling the digital components into an architecture that allows for control of each individual component as well as provides feedback through various power readings and spectral analysis all through a single connection to an external PC is a complex task. Updating the parameters of each component needs to be done quickly to ensure that during scenario executions they remain in sync with each other and to ensure that the central control entity does not get bogged down with updates in order to provide feedback to the user in a timely fashion. Also, the feedback data must be gathered quickly to ensure that the central control entity is available to receive user input in a timely fashion.

The use of a Microblaze processor along with its Fast Simplex Link (FSL) allows for the high speed transfer of information to and from the various components inside the FPGA. Each Microblaze processor allows for eight separate FSLs each with an input and output port. These ports allow for up to 32-bit data transfers. Many of the components along the data path are connected to the FSL to ensure high speed transfers–especially between the capture components which periodically grab 4096 samples for spectral analysis. The less timing critical components are tied to a simple On-chip Peripheral Bus (OPB). These connections can be seen in the block diagram in figure 6.2.

Many of the components in the system are discussed in great detail in chapter 3 and chapter 5. A few of the system components that haven't been mentioned include the internal UART, the Ethernet MAC, and the internal timer. The UART is used for low level communication to an external PC mostly for diagnostic purposes during development. The MAC is used to handle the communications through an Ethernet link. The timer is included to allow for periodic and time sensitive tasks to be performed, namely capturing data or updating the channel parameters. All three of these components require the processor to take action immediately. Thus, they are each tied to an interrupt controller which can be

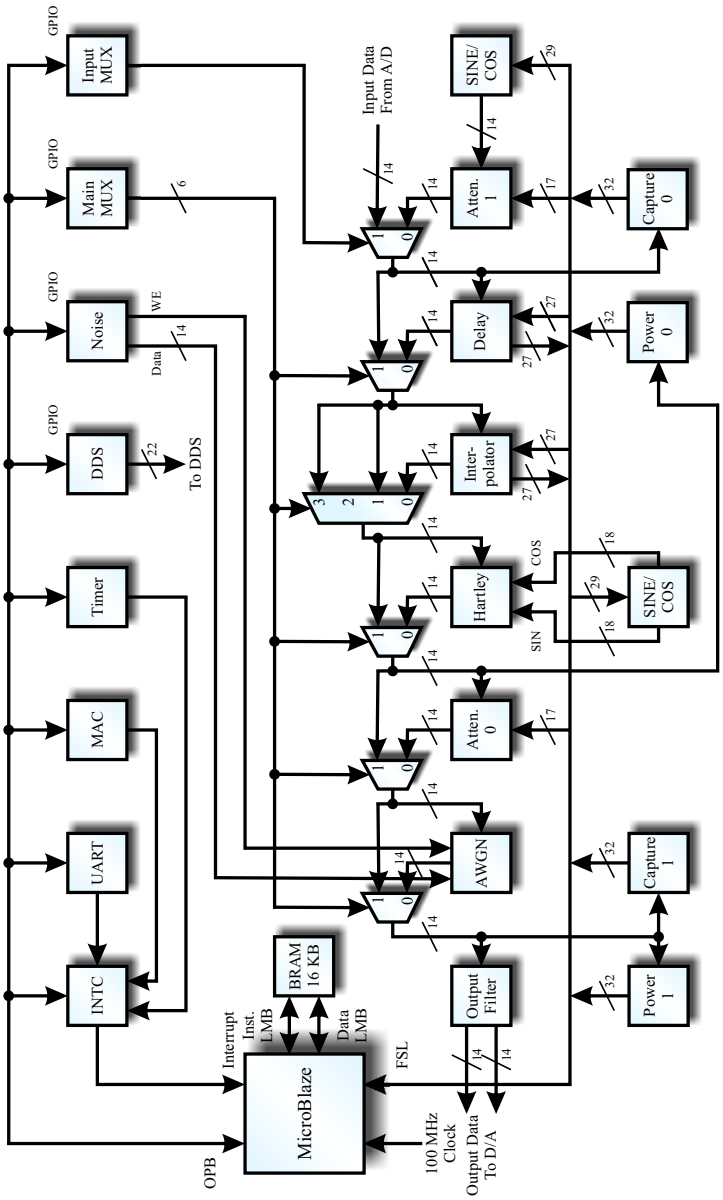


Figure 6.2: Block Diagram of the Digital Components.

queried to determine which component needs handling.

The code and data for the Microblaze processor must fit within the 16 KB block RAM (BRAM) used. While much of the code involved with executing a communications scenario can be contained in the user interface program on the PC, some rudimentary code must reside in the emulator to handle Ethernet communication, channel parameter updates, and feedback data gathering. Channel parameter updates require very little of either code or data, and the feedback data gathering components have their own BRAMs and registers to hold data. It is the Ethernet communication code that can be difficult to fit within the 16 KB allotment. Using the standard libraries for TCP/IP communication with the Microblaze required over 20 KB of code space. It was therefore necessary to write a very simple user datagram protocol (UDP) interface for the Microblaze. Handling only this simpler protocol produced Ethernet code below 1 KB.

6.2 Communicating with the User Interface

This section deals with the communication between the user interface and the various digital components in the emulator. The user interface must use standard units when dealing with channel parameters. How each component deals with the conversion from standard units to something meaningful to the hardware differs. The following subsections contain a brief discussion of the conversions implemented by the user interface for the main channel parameters.

6.2.1 Delay Settings

There are two main components used to implement the delay: the delay and the interpolator. The delay handles the individual sample jumps, while the interpolator allows for the fine tuning. The interpolator has two inputs: the offest, which sets the static delay, and the incrementor, which sets the dynamic

delay. It is the offset that is of interest in this section.

The delay is set in the unit of seconds. To convert that to samples, it is simply a matter of multiplying the desired delay by the sample rate. The integer portion of this is sent to the delay parameter. The decimal point portion must first be represented as a fraction of some value over 2^{27}, since there are 2^{27} different interpolation points between samples. This gives the following equations:

$$\text{delay} = \text{int(desired delay - 0.5)} \tag{6.1}$$

$$\text{interpolator}_{\text{static}} = \text{int}[2^{27}\text{decimal(desired delay} - 0.5) + 0.5] \tag{6.2}$$

6.2.2 Doppler Effect Settings

The Doppler effect, in the digital domain, requires correct settings to both the interpolator and single side-band modulation (SSB) components. This is shown in equation 3.14. The frequency dependent portion of this equation is handled by the interpolator, while the static shift portion is handled by SSB modulation. Defining the Doppler shift can be done in a number of ways. The most prevalent is by defining the relative velocity, v, between the transmitter and the receiver or by defining the center frequency shift. The latter of the two can be used to calculate the relative velocity fairly simply then only one set of equations needs to be used. It is important to remember that while the interpolator has 2^{27} discrete points, the sine and cosine function used to determine the SSB modulation frequency has 2^{29} discrete points.

$$\text{interpolator}_{\text{dynamic}} = \text{int}\left(2^{27}\,\frac{v}{c} + \text{sgn}(v)0.5\right) \tag{6.3}$$

$$\text{SSB} = \text{int}\left(2^{29}\,\frac{v}{c}\,56\text{MHz} + \text{sgn}(v)0.5\right) \tag{6.4}$$

6.2.3 Noise and Attenuation Settings

The key to determining the noise and the attenuation settings is first determining the respective power levels desired for the two. These levels can be determined quite easily if the SNR is known and the input and output power levels are known. Section 3.4.5 discusses the relationship between the input value of the AWGN generator and the output power level.

$$\text{noise} = \text{int}\left(2^{12}\ 10^{\text{power}_{\text{noise}}/10} + 0.5\right) \tag{6.5}$$

$$\text{attenuation} = \text{int}\left(2^{16}\ 10^{(\text{SNR}-\text{power}_{\text{in}}+\text{power}_{\text{noise}})/10} + 0.5\right) \tag{6.6}$$

Chapter 7

Conclusion and Future Work

7.1 Future Work

This thesis shows that functional requirements of the RF Channel Emulator have been met, yet there have been cases where measurements may not have been sufficiently accurate to be taken due to testing equipment limitations and where theory had difficulty explaining the measured results. This section focuses on those areas in more detail.

7.1.1 Phase Noise Measurements

Throughout the thesis, phase noise measurements were shown that proved to be much greater than the expected values. The test equipment used for this was an Agilent E4445A Spectrum Analyzer. While measuring the phase noise, and subsequently the jitter, of the various sampling clock sources in chapter 5, the results came to many orders of magnitude off of the expected values listed in the data sheets.

As discussed in section 3.6, phase noise is especially important to sampling clock signals as this noise gets incorporated into any sampled signal. This is due to the definition of phase noise itself: undesired fluctuations in the phase

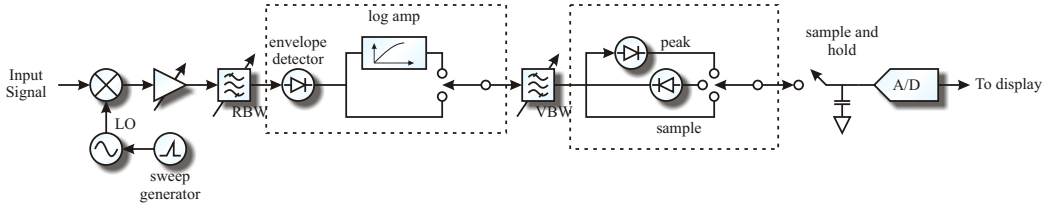


Figure 7.1: Simplified Spectrum Analyzer Block Diagram.

of a signal. This is in contrast to amplitude noise which will have little or no bearing on noise introduced into sampled signals due to clock noise since it is often the zero crossings of a clock signal that determine when sampling occurs. This illustrates the importance of being able to measure the phase noise exclusively as opposed to the total noise which includes any amplitude noise.

The use of a spectrum analyzer to measure phase noise does not give exclusively phase noise but rather measures the total noise. This is due to the fact that a spectrum analyzer gives the results in the magnitude of the power at particular frequencies with no means of determining the phase or amplitude of any other frequency within the spectrum. A block diagram of a typical spectrum analyzer is shown in figure 7.1. Here, it can also be seen that any noise measurements can only be measured down to the noise that will be introduced by the local oscillator of the analyzer itself. Many of the results given in this thesis are limited by this factor.

For future work, isolating the phase noise and accurately measuring it without being limited by the test equipment will give a much better understanding of the contribution of the clock phase noise into the system. For this work, the listed specifications for these measurements in the component data sheets show more than adequate performance.

7.1.2 Spurious Noise in the Clock Signal

The clock generated by the DDS, as discussed in section 5.1.5, has deterministic noise, or spurs in its frequency response seen in figure 5.9. At first glance these spurs appear to be what would be expected from quantization noise since the DDS has an internal 14-bit DAC. Quantization noise will fall on the harmonics of the fundamental frequency, 56 MHz. These harmonics will fold at half the internal clock frequency of the DDS, 200 MHz, and then again at DC. Determining the total number of possible harmonics is a fairly trivial task:

$$\text{Number of distinct harmonics} = \frac{\text{LCM}(200\text{MHz}, 56\text{MHz})}{56\text{MHz}} \tag{7.1}$$

$$= \frac{1400}{56} \tag{7.2}$$

$$= 25 \tag{7.3}$$

These 25 harmonics are listed in table 7.1. As it turns out, only three of these harmonics fall in the frequency range of figure 5.9 where more than seven spurs with magnitudes above -90 dBm can be seen. Although one of the spurs is attenuated in the final design through a bandpass filter, it would benefit the overall system to determine the source of the unaccounted for spurs in order to better reduce their effect on the system.

7.1.3 Sampling with Deterministic Clock Jitter

As discussed in section 3.6 and section 5.1.5, the assessment was shown that sampling a signal using a clock with phase noise will introduce noise into the sampled signal. In the particular case of spurious, or deterministic, phase noise, we are left without a specific representation of what the actual noise introduced into the sampled signal will be and therefore are left without a quantifiable value

Harmonic	Output Frequency	
f_0	56 MHz	*
f_1	112 MHz	
f_2	168 MHz	
f_3	176 MHz	
f_4	120 MHz	
f_5	64 MHz	*
f_6	8 MHz	
f_7	48 MHz	*
f_8	104 MHz	
f_9	160 MHz	
f_{10}	184 MHz	
f_{11}	128 MHz	
f_{12}	72 MHz	
f_{13}	16 MHz	
f_{14}	40 MHz	
f_{15}	96 MHz	
f_{16}	152 MHz	
f_{17}	192 MHz	
f_{18}	136 MHz	
f_{19}	80 MHz	
f_{20}	24 MHz	
f_{21}	32 MHz	
f_{22}	88 MHz	
f_{23}	144 MHz	
f_{24}	200 MHz	

Table 7.1: Harmonic Frequencies due to Quantization of the 56 MHz Sinusoid Generated by the DDS. '*' denotes frequencies that fall in figure 5.9.

for what the magnitude of deterministic phase noise can be to still meet noise requirements.

Section 3.6 draws a vague analogy that since the signal is essentially being multiplied by the clock signal in the time domain, it will be convoluted with the signal in the frequency domain. While this in itself is true, coming up with a frequency representation of impulse signals with periodic jitter can be somewhat problematic. We know through our discussion of phase noise in the previous sections of this chapter that the frequency representation of the clock signal by itself cannot be used in this matter given that this representation will contain both amplitude noise and phase noise. Once the phase noise is isolated, the initial convolution representation should hold fairly well.

For a simple demonstration of this, we can look at the clock signal with the spectral response shown in figure 5.9. Here, there are two main sinusoids of interest: one at 56 MHz and one at 44 MHz approximately 55 dB below the first. To generalize, we will approximate the clock signal as two sinusoids with different amplitudes and different frequencies. The relative phase can be disregarded since we are interested in the frequency response which covers all time, and there will always be at least one time where the two start in phase.

$$f_{\text{clock}}(t) = A_1 \sin \omega_1 t + A_2 \sin \omega_2 t \tag{7.4}$$

In order to separate the amplitude component, the following substitutions will be made:

$$A_3 = \frac{A_1 + A_2}{2} \tag{7.5}$$

$$A_4 = \frac{A_1 - A_2}{2} \tag{7.6}$$

$$\omega_3 = \frac{\omega_1 + \omega_2}{2} \tag{7.7}$$

$$\omega_4 = \frac{\omega_1 - \omega_2}{2} \tag{7.8}$$

Thus,

$$f_{\text{clock}}(t) = (A_3 + A_4)\sin[(\omega_3 + \omega_4)t] + (A_3 - A_4)\sin[(\omega_3 - \omega_4)t] \tag{7.9}$$

With some trigonometric identities, this can be reduced to:

$$f_{\text{clock}}(t) = 2A_3 \sin\omega_3 t \cos\omega_4 t + 2A_4 \cos\omega_3 t \sin\omega_4 t \tag{7.10}$$

Using the following relationship,

$$A\ \sin\alpha + B\ \cos\alpha = \sqrt{A^2 + B^2}\sin\left[\alpha + \arctan\left(\frac{A}{B}\right)\right], \tag{7.11}$$

equation 7.10 can be reduced to:

$$f_{\text{clock}}(t) = 2\sqrt{A_3^2\cos^2\omega_4 t + A_4^2\sin^2\omega_4 t}\ \sin\left(\omega_3 t + \arctan\frac{A_4\sin\omega_4 t}{A_3\cos\omega_4 t}\right) \tag{7.12}$$

We can now clearly see a distinction between the time dependent amplitude of this function and the time dependent frequency or phase of this function. Since our goal was to remove the amplitude portion of the clock function, we will do so in the next step. It is also important to represent the phase in terms of ω_1 given that in our particular case, A_1 is much greater than A_2.

$$\begin{aligned}
f'_{\text{clock}}(t) &= \sin\left(\omega_3 t + \arctan\frac{A_4}{A_3}\tan\omega_4 t\right) && (7.13)\\
&= \sin\left(\omega_3 t + \arctan\frac{\tan\omega_4 t + x}{1 - x\tan\omega_4 t}\right), \\
&\quad x = (\frac{A_4}{A_3} - 1)\frac{\tan\omega_4 t}{1 + A_4/A_3\tan^2\omega_4 t} && (7.14)\\
&= \sin\left[\omega_3 t + \arctan(\tan\omega_4 t) + \arctan x\right] && (7.15)\\
&= \sin\left[(\omega_3 + \omega_4)t + \arctan(\frac{A_4}{A_3} - 1)\frac{\sin\omega_4 t\cos\omega_3 t}{\cos^2\omega_4 + A_4/A_3\sin^2\omega_4 t}\right] && (7.16)\\
&= \sin\left(\omega_1 t - \arctan\frac{A_2\sin(\omega_1 - \omega_2)t}{A_1 + A_2\cos(\omega_1 - \omega_2)t}\right) && (7.17)
\end{aligned}$$

Equation 7.17 is in the form we were looking for. This form, by itself, doesn't lend itself well to the Fourier Transform. But, taking some approximations given the assumption that A_1 is much greater than A_2 yeilds:

$$f'_{\text{clock}}(t) \approx \sin\left(\omega_1 t - \frac{A_2}{A_1}\sin(\omega_1 - \omega_2)t\right) \tag{7.18}$$

This is much easier to transform into the frequency domain:

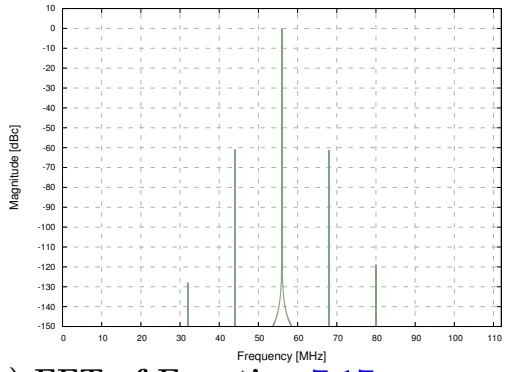
$$|F'_{\text{clock}}(\omega)| \approx \sum_{n=-\infty}^{\infty} J_n(A_2/A_1)\delta[\omega - \omega_1 - n(\omega_1 - \omega_2)] \tag{7.19}$$

Since we are really interested in a rule of thumb regarding the magnitude of the additional spectral components in a signal sampled with a noisy clock, we can approximate even further with the same assumption that $A_1 >> A_2$.

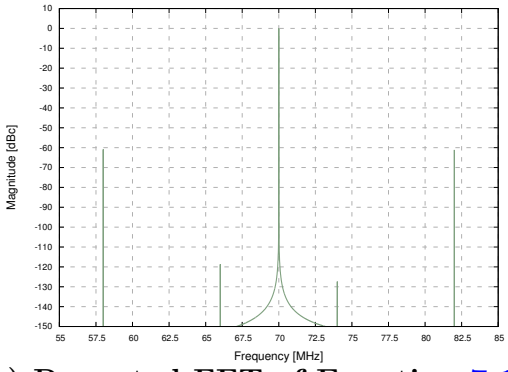
$$\begin{aligned} |F'_{\text{clock}}(\omega)| \quad &\approx \quad \delta(\omega - \omega_1) + \frac{A_2}{2A_1}\delta(\omega - \omega_1 \mp \Delta\omega), \\ &\Delta\omega = \omega_1 - \omega_2 \end{aligned} \tag{7.20}$$

Equation 7.20 gives a very manageable representation of what can be expected with given monochromatic, deterministic jitter on a clock signal. Although it is difficult to calculate the accuracy of this approximation precisely, we can run a simulation to show how it holds up to the data acquired. For this, we'll use the coefficients from our testing: $A_1 = 1.0$, $A_2 = -55dBc$, $\omega_1 = 2\pi 56MHz$, and $\omega_2 = 2\pi 44MHz$. The results of this simulation can be seen in figure 7.2. This shows the FFT of equation 7.17 along with the convolution of the FFT with a digital 70 MHz sinusoid. For comparison, these plots are also shown with the approximation in equation 7.20. As we can see, the approximation holds well for the three most dominate frequencies while it overlooks all the rest.

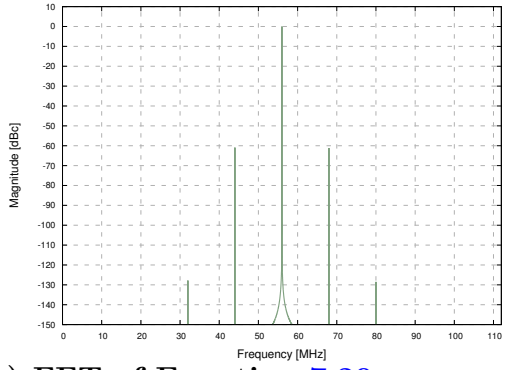
For further comparison, the positive-slope, zero crossings of equation 7.10 were found using Newton's method for finding roots. These points were used to sample a 70 MHz sinusoid. Displayed in the figure is the FFT of these sampled



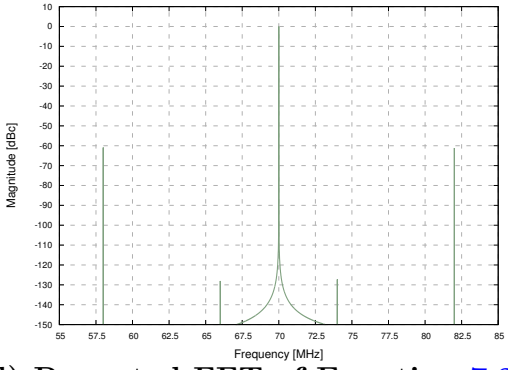
a) FFT of Equation 7.17.



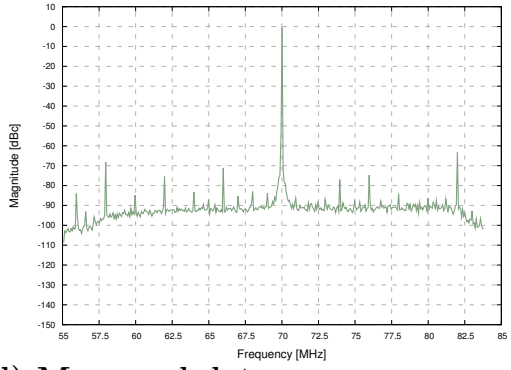
b) Repeated FFT of Equation 7.17.



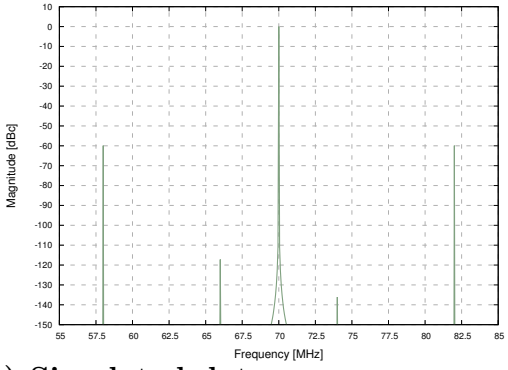
c) FFT of Equation 7.20.



d) Repeated FFT of Equation 7.20.



d) Measured data.



e) Simulated data.

Figure 7.2: Simulation Results for Sampling using a Clock Signal with Monochromatic Deterministic Phase Noise.

points to give a simulation of what, theoretically, the frequency response should resemble. As we can see, both the previous methods that assumed the convolution of the frequency response of the clock signal with only phase noise and the frequency response of the input signal gave very good approximations for the locations and magnitudes of all frequencies over -80 dBc. For a final comparison, the experimental data is also shown. Keep in mind that our discussion only incorporated monochromatic deterministic phase noise, while the actual clock signal included many different spurs of different frequencies and magnitudes.

7.2 Conclusions

This thesis presents the overall system performance characterization of the RF Channel Emulator. An emphasis of the digital component design and analysis is presented in this work as much of the input and output analog stages were characterized by Kyle Woolrich [22]. Due to the necessity to improve performance, some of the original analog design was modified and analyzed here. From the component level to the system level, the RF Channel Emulator performs very well across the specified band in emulating channel characteristics without significantly degrading the input signal.

The use of digital components proved to be an effective method to implement channel emulation parameters. Implementing a significant delay to the signal is only practically possible in a lab environment through the use of digital delay. By itself, the memory controller allows for as large of a delay as there is RAM to store it with the ability to change the delay rapidly and dynamically. Digital interpolation allows the delay to be fine tuned to much less than a single sample, which is an inherent limitation of a digital delay. The interpolation component also allows for rapid delay changes in addition to providing a means for improved Doppler emulation through data "throttling" while minimizing impact to signal integrity with its very linear phase and flat magnitude in-band response. Wool-

rich [22] showed that single side-band modulation can be effectively achieved in the analog output stage of the emulator design with the inclusion of a mixer and an additional DDS. As with the addition of any analog component, additional thermal noise and inherent nonlinearities can increase the overall random and deterministic noise respectively. A digital implementation, as presented in this thesis, can eliminate the need for additional analog components while providing a low noise response. Finally, the digital additive noise component produces a white (flat) frequency response with a Gaussian probability density function without the need for additional analog components.

Despite being designed around specific channel requirements such as center frequency and bandwidth, the RF Channel Emulator has a robust and flexible design that may be modified to fit a wide range of specifications with minimal work. The design of the digital delay can accommodate many different DRAM configurations with very few changes to the hardware design code allowing for significant increases in the size of the RAM module capacity which translates directly to an increase in the maximum delay possible. The other digital components are all designed with highly parallel topologies allowing for increases in the data rate to meet other bandwidth requirements. In addition, the digital components were all designed in VHDL which reduces overhead in porting to another hardware platform if necessary for a significant increase in performance requirements. The physical hardware is composed of modular, off-the-self components easing the ability to change or replace a single component with minimal impact to the rest of the system.

Although much design effort went into eliminating noise sources and meeting design requirements set by JPL, there are inherent limitations in the RF Channel Emulator. The most prevalent limitation is in the maximum delay possible. While the requirement of two seconds was exceeded, this only allows for the emulation of systems operating roughly as far as twice the distance to the moon. Many of the missions operated by JPL work at much greater distances. The delay

limitation also affects the duration for which certain scenarios can be executed. For example, emulating the reception of transmissions from the Earth to a satellite in geosynchronous orbit would effectively require a continuous Doppler shift due to the combined effects of special and general relativity. Since the Doppler Effect is emulated using a continuous change in the delay, the time of the emulation is limited by how long the delay buffer would take before it was overrun. Although, this limitation is rendered moot since the complexities of simultaneity in relativistic scenarios require both the change in apparent distance to be non-zero while the round trip propagation time to be constant which, outside of slowing time down in a portion of the lab, is impossible to emulate. It should also be noted that a geosynchronous orbit around the Earth would require a time scaling factor of roughly $1 + 2^{-31}$ while the interpolation hardware can currently only achieve an accurate scaling factor down to $1 + 2^{-27}$.

Aside from the limitations with the delay hardware, there are inherent limitations to the achievable signal to noise ratio with the current design. The dominant contributor to the spurious noise in the system is the DDS. While the input reference oscillator has an impeccably clean frequency response, the output of the DDS is impacted by spurs associated with quantization and nonlinearities in the internal DAC of the DDS itself. Since the DDS is used to generate the sampling clock, these spurs are incorporated into the digital signal reducing the overall SNR. Although these spurs will be a component of any DDS, using an industrial grade DDS could reduce their magnitude.

Overall, the RF Channel Emulator provides a platform for accurately emulating channel characteristics while giving the ability to instantly change static or dynamic parameters of the channel. It is my expressed hope that the RF Channel Emulator will ease performance characterization of communication protocols used by JPL on current and future missions.

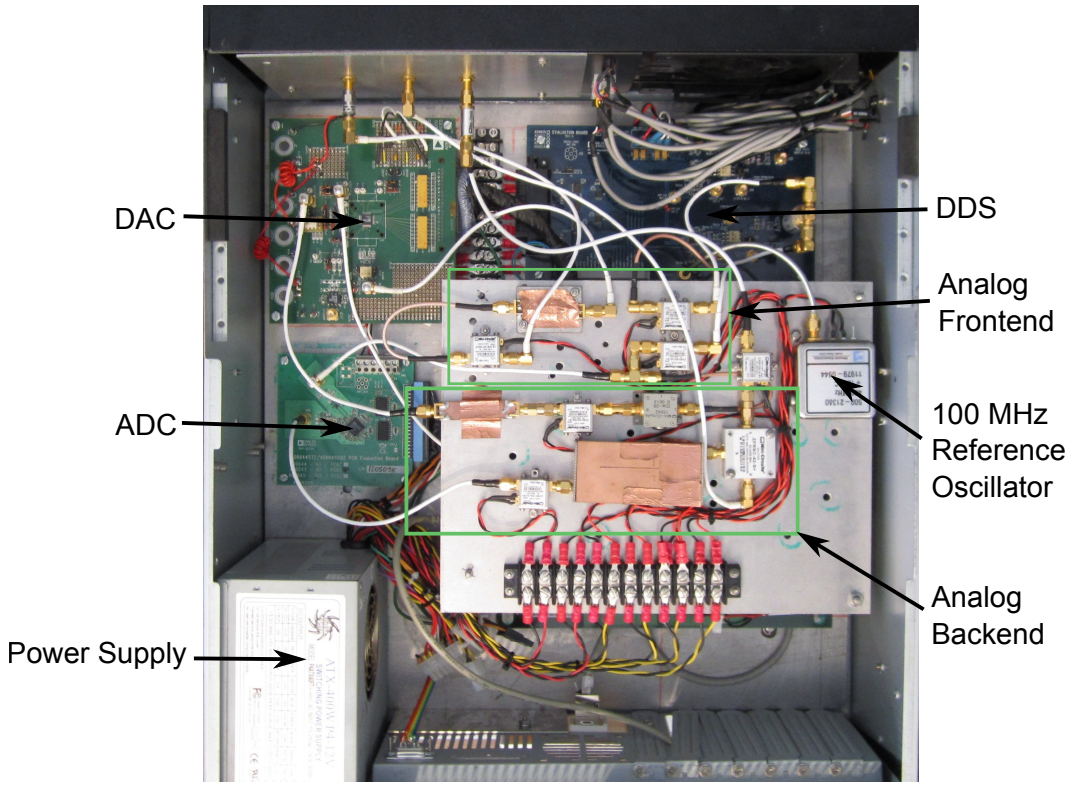


Figure 7.3: Final Assembly. The final hardware implementation assembly was constructed in a rack mountable frame.

Bibliography

[1] Additive white gaussian noise (awgn) core v1.0 data sheet. www.xilinx.com/support/documentation/ip.../awgn.pdf, October 2002.

[2] Sle700 satellite link emulator data sheet. http://www.dbmcorp.com/pdf/sle700.pdf, November 2002.

[3] Fading in wireless communications. Wireless Technology (YlessTech), October 2006. http://www.ylesstech.com/terminology.php?letter=all&id=15.

[4] Mixers. Microwaves101.com, August 2006. http://microwaves101.com/encyclopedia/mixers.cfm.

[5] Virtex-ii pro and virtex-ii pro x platform fpgas: Complete data sheet. November 2007.

[6] Ashok Ambardar. *Analog and Digital Signal Processing*. PWS Publishing Company, Boston, USA, 1995.

[7] C. R. Baugh and B. A. Wooley. A two's complement parallel array multiplication algorithm. *IEEE Trans. Comput.*, 22(12):1045–1047, 1973.

[8] A.D. Booth. A signed binary multiplication technique. *Quart. Journ. Mech. and Applied Math.*, 4(2):236–240, 1951.

[9] Emmanuel Boutillon, Jean-Luc Danger, and Adel Ghazel. Design of high speed awgn communication channel emulator. *Analog Integr. Circuits Signal Process.*, 34(2):133–142, 2003.

[10] I. A. Glover and P. M. Grant. *Digital Communications, 2nd Edition*. Prentice-Hall, New Jersey, USA, 2003.

[11] Guillermo Gonzalez. *Microwave Transistor Amplifiers: Analysis and Design, 2nd Edition*. Prentice-Hall, New Jersey, USA, 1996.

[12] Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan. *Simulation of Communication Systems: Modeling, Methodology, and Techniques*. Kluwer Academic/Plenum Publishers, New York, NY, USA, 2000.

[13] J.F. Kaiser. Nonrecursive digital filter design using i0-sinh window function. *IEEE Int. Symposium on Circuits and Systems*, pages 20–23, April 1974.

[14] Walt Kester. *Analog-Digital Conversion*. Analog Device, Inc., USA, 2004.

[15] Walt Kester. Converting oscillator phase noise to time jitter. *Analog Devices*, October 2005.

[16] C. Kurth. Generation of single-sideband signals in multiplex communication systems. *Circuits and Systems, IEEE Transactions on*, 23(1):1–17, Jan 1976.

[17] H. Scheuermann and H. Gockler. A comprehensive survey of digital trans-multiplexing methods. *Proceedings of the IEEE*, 69(11):1419–1450, Nov. 1981.

[18] M.J. Schulte and J.E. Stine. Symmetric bipartite tables for accurate function approximation. In *Computer Arithmetic, 1997. Proceedings., 13th IEEE Symposium on*, pages 175–183, Jul 1997.

[19] Bernard Sklar. Rayleigh fading channels in mobile digital communication systems part i: Characterization. *Communications Magazine*, Jul 1997.

[20] Bernard Sklar. *Digital Communications: Fundamentals and Applications*. Prentice-Hall, New Jersey, USA, 2001.

[21] C. Tseng and S. Pei. Design of discrete-time fractional hilbert transformer. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 5, pages 525–528 vol.5, 2000.

[22] Kyle Woolrich. Emulating a space communication channel using analog and digital signal processing. Master's thesis, California Polytechnic State University, San Luis Obispo, June 2008.

Appendix A

Statement of Work

70 MHz Channel Emulator Specification

Background

Evaluation of end-to-end communications systems in use for space exploration often requires emulation of the RF path. While noise and delay may be adequately simulated in a Linux environment for the evaluation of their effects on communications protocols, these simulations are inadequate for the purpose of testing systems with actual radio hardware. To adequately emulate the space link channels, an emulator is required that can inject realistic noise, delay and Doppler effects into an RF link between radios under test.

Lunar exploration will likely rely on radio links at S-band, Ka-band, frequencies that are impractical to directly digitize and digitally process in the context of a link channel simulation; in addition, the data rates required range from 72 Kbps to 25 Mbps. For this research project, as a proof-of-concept for the digital processing techniques required, operation at lower intermediate frequencies and somewhat lower maximum data rates will be required. Since lab UHF radios are available that operate at 70 MHz IF and maximum data rates of 4 Mbps, these parameters were selected for this emulator.

Summary Description

The emulator shall use digital techniques, and provide for digital control from an external PC using a Graphic User Interface such as LabView (to be negotiated). The Digital Link Channel Emulator (DLCE) shall be used in a lab test environment shown in figure 1:

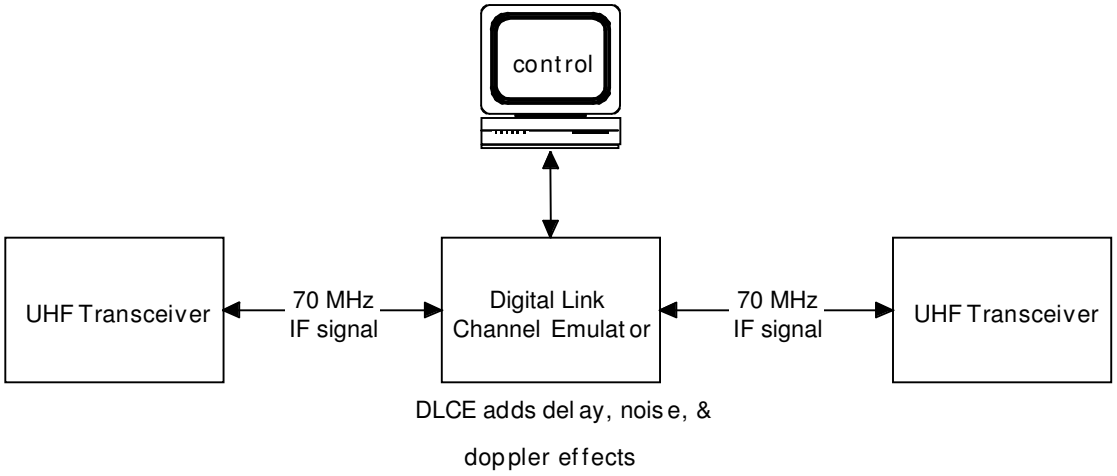


Figure 1: DLCE in Lab Test

The specifications below are a combination of firm requirements and some performance goals that will be negotiated; should initial work (or budget) prohibit the achievement of the stated goals, the design should be such that future modifications and improvements are not precluded.

Interface Description

The input and output characteristics of the RF signal in and out of the channel simulator are as follows:

70 MHz center frequency without doppler
0 dBm input power maximum; normal input signal operating range -7 to -12 dBm.
Output power at 70 MHz = -12 dBm

Modulation types (SN formatting representative of current baseline Lunar radio requirements)

suppressed carrier, uncoded at 4 Mb/s (for an 8 MHz bandwidth)*
residual carrier, coded at 72, 192 or 256 Kb/s

* Note: The current Lunar baseline calls for 25 Mbps uncoded suppressed carrier, but for the purposes of this initial link channel simulator a 4 Mb/s capability will be the design point. The task should include consideration of the technology required to achieve this higher bandwidth in future designs.)

Functional Objectives & Requirements

The DLCS is a hybrid analog/digital device to simulate the RF space path of a radio signal being used for communication between two spacecraft. To accomplish this, a 70 MHz intermediate frequency can be used between two standard lab test radio transceivers, with the signal being digitized, manipulating the digital signal with appropriate noise, frequency shift and amplitude changes, and the resulting signal upconverted to 70 MHz for insertion into a receiver baseband processor.

Typical Utilization of the DLCS

A typical radio test scenario would be to simulate the radio link characteristics between a spacecraft and a lander, with the spacecraft passing overhead at 300 km typical orbital altitudes over a period of 10 minutes (number approximate.) This results in a signal that fades in and out as a function of antenna pattern and relative geometry, possible multipath, noise and Doppler frequency shift during the pass as the geometry changes. Another possible scenario might be communications between two spacecraft as they approach for a rendezvous; if one was at lunar distances and the other was enroute to earth, propagation delays might be on the order of 1-2 seconds.

The specifications below represent the performance and design goals for the CalPolySLO simulator:

Simulator Control

The simulator shall be required to accept continuous control inputs to determine the amount of delay, fading, noise and Doppler characteristics using an Ethernet interface to a control computer based on manual operator inputs or a script. A GUI is desired and optimally will be based on LabView. GUI or operator control shall provide for steady-state selection of Doppler, Eb/No and delay, as well as providing for dynamic control of those parameters via script or other program input (e.g. for use in piping the output of an orbital simulation running in the control computer to the channel simulator.)

Digitization and Sampling

It is desired that the digital signal domain be in both I and Q so that noise and fading in these channels may be either independent or correlated.

(For this the recommended digitization process is that the simulator should sample at a 56 MHz rate using 12-bit digitization for a 14 MHz digital IF, which is then complex basebanded to DC for ease of subsequent computation.)

Delay Processing

Sufficient memory and buffering should be provided for a variable propagation delay from 0 to 2 sec (desired), with 1 msec granularity. (Delay processing shall accommodate a minimum of 500 msec delay)

Noise Processing and Signal Fading

The simulator should add noise as complex Gaussian. The simulator should be capable of adding independent signal fading to the I and Q channels. Typically the mobile satellite fading environment stastistics follow a complex Ricean distribution, essentially a Rayleigh distribution plus a line-of-sight component. The signal fading may be accomplished by either real time generation of fading components or reading the I – Q fading components from a stored file.

In order to accommodate the operation of the projected radios, the emulator shall mimic the signal as seen by the baseband processor after the front end ACG process is applied. This radio AGC system provides the baseband processor with a -12 dBm signal+noise and the effect of path noise, system noise and signal amplitude fading will be to change the Eb/No signal-to-noise ratio. The simulator output should leave the input data signal at the nominal -12 dBm level after the delay and doppler processing, with the effect of noise and fading being that the Eb/No changes from 0 dBm (noise at signal level) down to zero noise added (with 10 dB Eb/No being a typical operating point).

It is desireable to operate the noise and fading signal components on I and Q channels independently. JPL will collaborate with CalPoly SLO on potential noise injection designs.

Doppler Processing

The DLCS shall impress user specified Doppler and Doppler rate upon the output frequency as follows:

Doppler shift of +/- 3.5 KHz at 70 MHz (to be negotiated, TBN)
Instantaneous Doppler rate of 1000Hz/sec (TBN) to 200 Hz/sec (TBN)

70 MHz Channel Simulator Specifications

According to the attached proposal from Ed Satorius, you are prepared to build a programmable delay line with the following specs:

In/Out frequency = 70 MHz
8 bit A/D and D/A
FPGA input data rate = 1.6 Gbps
SRAM Memory = 20 MBytes

I would like to modify these specs as follows:
In/Out Frequency = 70 MHz or 390-450 MHz
Variable delay from 0 to 2 sec
Impress user specified Doppler and Doppler rate upon the output frequency:
Doppler shift of +/- 3.5 KHz at 70 MHz or +/- 20 KHz at 400 MHz
Instantaneous Doppler rate of 1000Hz/sec (TBC) to 200 Hz/sec (TBC)

JPL

INTEROFFICE MEMORANDUM
August 16, 2005

TO: Caroline Racho
FROM: Ed Satorius
SUBJECT: Architecture for the Channel Simulator

A block diagram of the simulator testbed is presented in Figure 1.

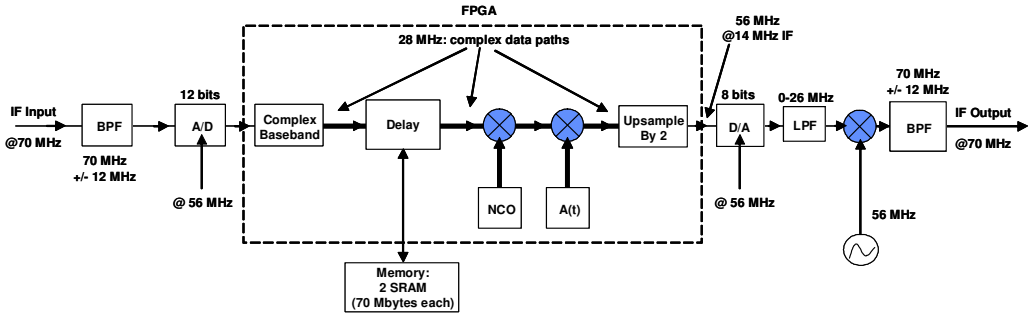


Figure 1. Simulator block diagram.

Note that bandpass sampling is used to shift the 70 MHz input IF down to a digital IF of 14 MHz, which is then complex basebanded (to DC). The digital complex baseband block diagram is depicted in Figure 2.

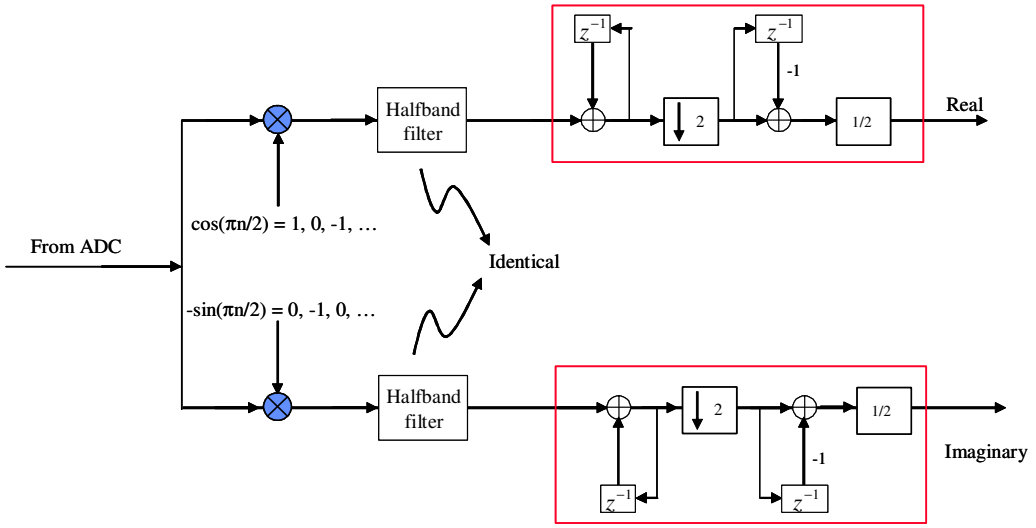


Figure 2. Digital complex baseband down-conversion and decimation.

The digital complex baseband down-conversion scheme is chosen for its computational efficiency and flexibility [E. Satorius, "Candidate MCAS receiver front-end architectures and issues," JPL Internal Memorandum, 27 April 1998]. Although the depiction in Figure 2 shows that the digital mixing is accomplished using two multipliers, a counter and two 4-to-1 multiplexers are used in the actual implementation. The counter controls the selection for the multiplexers. The multiplexers receive their inputs from identical sources but via different input channels. Two of the inputs to the multiplexers are tied to zeros. The other two are tied to the incoming A/D data and its inverted version.

Following digital mixing, a pair of identical half-band filters are applied to remove aliased images near the Nyquist band edge. A 10-tap FIR half-band filter is chosen for this purpose. The FIR half-band filter has the following transfer function:

$$H(\omega) = (2^{-8} + 2^{-9})(1 + e^{-10j\omega}) - (2^{-5} + 2^{-6} + 2^{-9})(e^{-2j\omega} + e^{-8j\omega}) + (2^{-2} + 2^{-5} + 2^{-7} + 2^{-8})(e^{-4j\omega} + e^{-6j\omega}) + 2^{-1}e^{-5j\omega}$$

Note that although the filter is 10-th order, only 4 multiplications are required for its implementation. These multiplications can be implemented using bit shifts and adds because the non-zero filter coefficients are given by:

$$h(0) = 2^{-8} + 2^{-9}; h(2) = -(2^{-5} + 2^{-6} + 2^{-9}); h(4) = 2^{-2} + 2^{-5} + 2^{-7} + 2^{-8}; h(5) = 2^{-1}$$

Following the half-band filters, the (complex) data are further decimated by 2 using sum-and-dump filters (creating one output from the mean of two successive input samples). At this point the complex data are output from the complex basebander at 28 MHz (both real and imaginary data channels).

With reference to Figure 1, the complex data are delayed (variable delay up to 2 seconds using dual, 70 Mbyte SRAMs), frequency and amplitude shifted to simulate the effects of both time-varying multipath and Doppler offsets. Following the amplitude shift, the data are then mixed back up to the 14 MHz digital IF and upsampled back to 56 MHz. A block diagram depicting this process is presented in Figure 3. Note that the upsampler (duplication of each input sample) is essentially a digital zero-order-hold.

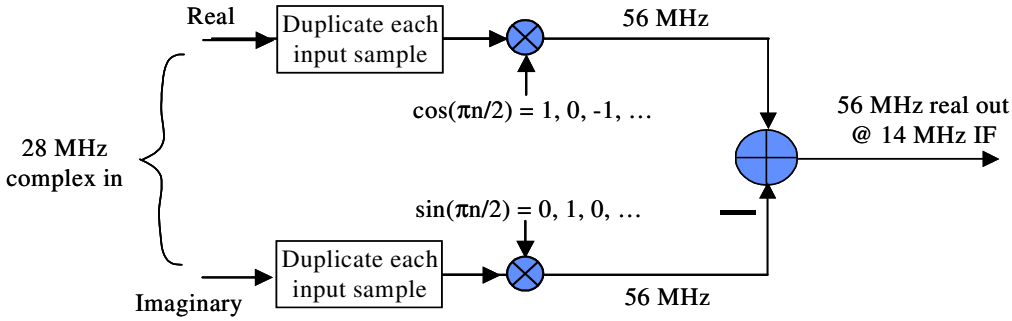


Figure 3. Digital complex baseband up-conversion and upsampling.

Following upsampling to a 14 MHz digital IF (Figure 1), the digital data stream is converted back to analog via a D/A and lowpass filter and finally mixed up to the final 70 MHz output IF. A sample simulation example is provided in Figure 4 corresponding to a 4 Mbps BPSK signal.

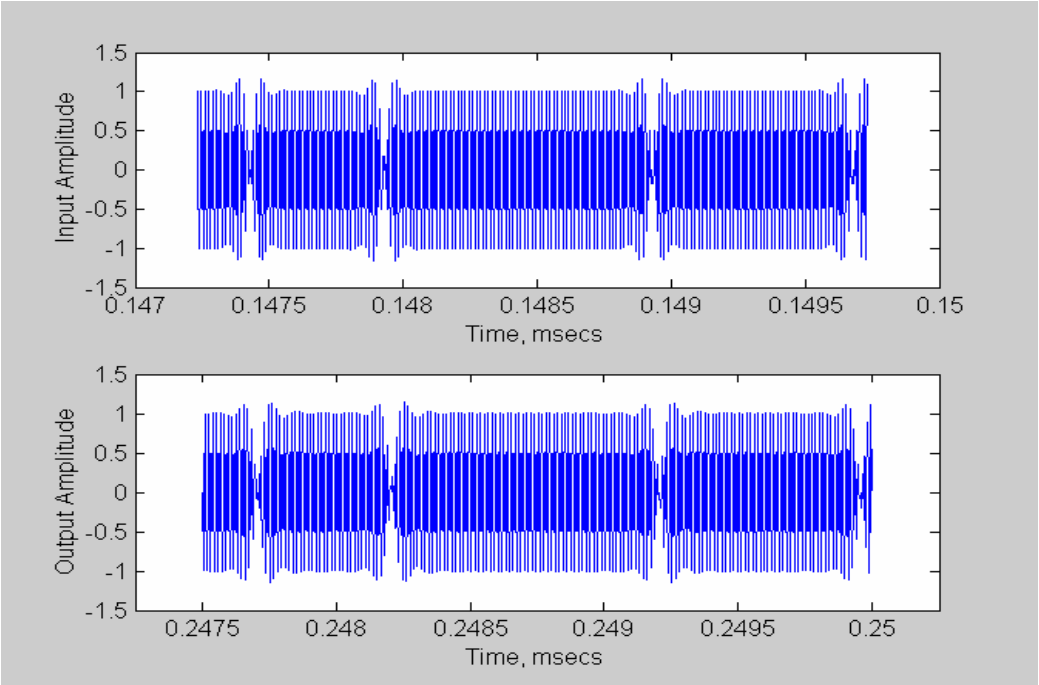


Figure 4. Matlab simulation example: 100 μsec delay simulated without Doppler or amplitude shift and without the post D/A analog lowpass filter in Figure 1.

The input waveform (top plot in Figure 4), is immediately after the input 70 MHz bandpass filter in Figure 1 (prior to the A/D) whereas the output (bottom plot in Figure 4), immediately follows the output 70 MHz bandpass filter in Figure 1 (after the mixer). The total synthesized delay between the input and output waveforms in this example (including the delay added by the outbound 70 MHz bandpass filter) is approximately 100 micro secs. As is seen from Figure 4, the digital processing in the FPGA does not add significant distortion to the output waveform.