LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TH-224428

# Under-sampling in a Multiple-Channel Laser Vibrometry System

J. Corey

September 14, 2006

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

# Under-Sampling in a Multiple-Channel Laser Vibrometry System

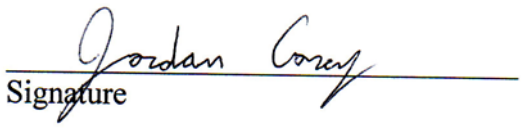
A Thesis
Presented to the Faculty of
California State Polytechnic University
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

By
Jordan Corey
Lawrence Livermore National Laboratory
UCRL-TH-224428
March 2007

**AUTHORIZATION FOR REPRODUCTION**
**OF MASTER'S THESIS**

I grant permission for the reproduction of this thesis in its entirety or any of its parts, without further authorization from me.



Signature

3/7/2007

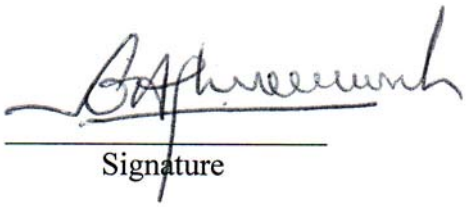
Date

**APPROVAL PAGE**

TITLE: Under-Sampling in a Multiple-Channel Laser Vibrometry System

AUTHOR: Jordan Corey

DATE SUBMITTED: March 7, 2007



Dr. Samuel Agbo
Adviser or Committee Chair

Signature

Dr. Bryan Mealy
Committee Member

Signature

Dr. Dennis Derickson
Committee Member

Signature

## ABSTRACT

### Under-Sampling in a Multiple-Channel Laser Vibrometry System

### Jordan Corey

Laser vibrometry is a technique used to detect vibrations on objects using the interference of coherent light with itself. Most vibrometry systems process only one target location at a time, but processing multiple locations simultaneously provides improved detection capabilities. Traditional laser vibrometry systems employ over-sampling to sample the incoming modulated-light signal, however as the number of channels increases in these systems, certain issues arise such a higher computational cost, excessive heat, increased power requirements, and increased component cost.

This thesis describes a novel approach to laser vibrometry that utilizes under-sampling to control the undesirable issues associated with over-sampled systems. Under-sampling allows for significantly less samples to represent the modulated-light signals, which offers several advantages in the overall system design. These advantages include an improvement in thermal efficiency, lower processing requirements, and a higher immunity to the relative intensity noise inherent in laser vibrometry applications. A unique feature of this implementation is the use of a parallel architecture to increase the overall system throughput. This parallelism is realized using a hierarchical multi-channel architecture based on off-the-shelf programmable logic devices (PLDs).

## ACKNOWLEDGEMENTS

I would like to thank everyone at the Lawrence Livermore National Laboratory for making this project possible. Thank you to my project supervisor, Dr. Liesl Little, for helping guide me throughout the entire project on all the technical aspects of the system. Also, thank you to my administrative supervisor, Bruce Henderer, for giving me the opportunity to work at the lab, and keeping the CalPoly / LLNL co-operative thesis program running. I'd also like to thank everyone in the Instrumentation Systems Group for taking the time to help support me on various occasions throughout the project.

I'd like to thank my committee chair, Dr. Samuel Agbo, for his suggestions and supervision of my project, as well as Dr. Bryan Mealy, for providing additional feedback about the thesis.

Lastly, I'd like to thank my family for supporting and encouraging me throughout both my undergraduate and graduate schooling and research.

# Table of Contents

# List of Figures

# I. Introduction

The purpose of this project is to create an under-sampling system to measure the mechanical vibrations that are present on a given target. In many cases, these vibrations are extremely small in amplitude (on the order of a nanometer) and difficult to detect. By employing laser vibrometry technology, these vibrations can not only be identified, but can be detected from significant distances.

Classical vibrometry systems only used a single channel and the incoming vibration data could be easily over-sampled. However, newer systems may employ many spatial channels to improve detection capabilities. These channels are composed of independent laser beams that reflect off the target of interest in multiple physical locations. In these cases, under-sampling can produce superior results to over-sampling and include many other benefits. This thesis describes the novel approach of using under-sampling within a multiple-channel laser vibrometry system. A literature search shows no pre-existing vibrometry systems with the set of attributes presented in this thesis.

# II. Background

## *Laser Vibrometry*

Laser vibrometry originated in the 1970's. The main concepts of laser vibrometry are based on the theories of speckle metrology developed in the 1960's that state that when a coherent beam of light hits a surface that is considered to be optically rough, the reflected wavelet components become out of phase with each other. An optically rough surface is a surface whose roughness is much larger than the wavelength of the incoming light. These coherent wavelets that are no longer in-phase, interfere with each other constructively and destructively to produce areas of high and low intensity which appear as a "speckle" pattern.

Early researchers believed that the speckle pattern was detrimental to their study of lasers and they tried to eliminate it from occurring, especially in the area of holography [19]. Eventually, techniques such as laser speckle photography showed that the speckle could be advantageous in laser metrology. One of the first experiments done involving laser metrology was an attempt to measure the rotation speed of a wheel by measuring the phase change of the scattered light [22]. The experiment worked by splitting a coherent laser into reference and signal beams. The signal beam was then reflected off the target (rotating wheel) and shifted in frequency due to the Doppler Effect. The reflected light then recombined with the reference beam incident on a photo-detector. When the coherent light beams recombined, they produced a beat frequency which could be detected and then processed to extract the velocity information. Figure 1 shows a

diagram of the experiment. Using the interference of scattered light to measure a velocity is known as laser velocimetry, which is a subset of laser metrology.

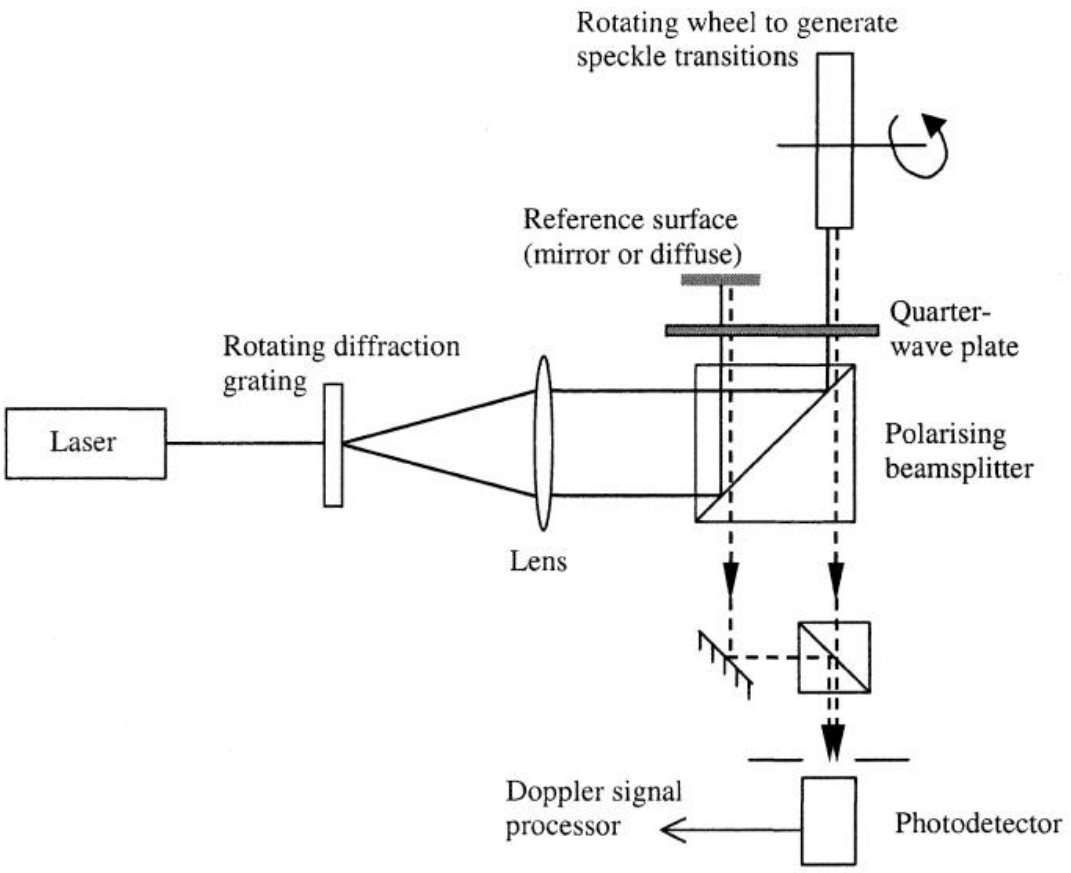



**Figure 1 - Diagram of Laser Velocimetry Experiment [22]**

Laser vibrometry is used when measuring translational position vibrations rather than velocity measurements. When a laser beam hits a target and is reflected, the laser light becomes phase-modulated by the target. This modulated light is then combined with a coherent reference source incident on a photo-detector in heterodyne detection. A beat frequency is produced in the output current of the photo-detector which is then

processed to extract vibration information. Possible applications of laser vibrometry are a contemporary field of study that has continued to expand into the current decade [22].

### *Under-sampling*

Under-sampling is a technique that at first may seem to be a strictly theoretical concept, but in reality it serves a number of useful purposes. Under-sampling is commonly used in radio frequency (RF) test equipment and communications systems [11]. Generally, under-sampling is employed when over-sampling is not a feasible solution and other methods for frequency translation, such as RF mixing, are prohibitive for a variety of reasons (size, cost, power, etc.).

Due to the limitations of current analog-to-digital converter (ADC) technology, the sampling rate of oscilloscopes today is limited to several gigahertz. This restriction requires that any observable signal needs to be less than half this frequency. Oscilloscopes that operate at high speeds are also extremely expensive. However, if the signal of interest is more than a couple gigahertz, an over-sampling scope will not work, because the higher frequency data is filtered out by an anti-aliasing filter before the ADC to prevent it from aliasing onto lower frequency components.

To remedy this frequency limitation, there are a number of under-sampling oscilloscopes available that divide the incoming signal into frequency regions and band-pass filter them [23]. The oscilloscopes then sample each region using under-sampling to record the higher frequency data with a lower sampling rate. Using this technique, oscilloscope bandwidth can range up to eighty gigahertz and above, which is much greater than any type of over-sampling scope can provide.

Modern digital receivers also commonly use under-sampling [28]. The goal of most digital receivers is to receive and demodulate a signal that is transmitted with a carrier frequency of anywhere from a few megahertz up to tens of gigahertz. However, the bandwidth of the actual data riding on the carrier is usually small compared to the carrier itself. There are several methods to extract the signal information from the carrier.

If the carrier frequency is low enough, the incoming signal can be over-sampled and then processed in the digital domain to extract the data of interest. In traditional systems, the incoming RF signal is processed with an analog mixer to convert it down to baseband, or more commonly, to an intermediate frequency (IF) [12]. The IF signal is then filtered and amplified before it is mixed down to baseband. The disadvantages of this analog mixing technique are that the RF and IF mixers can be bulky, expensive, thermally inefficient, and require a large amount of power.

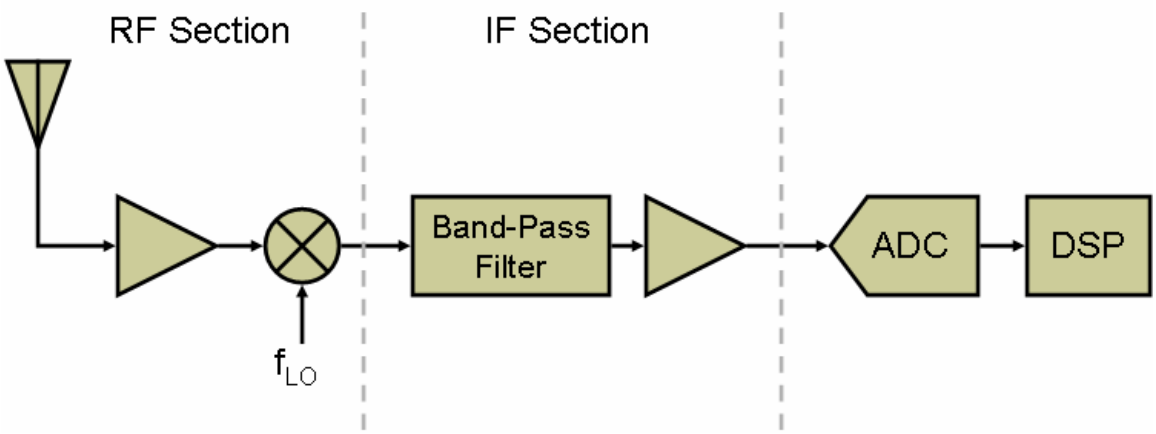



**Figure 2 - Software Radio [12]**

In recent years, digital receiver technology has started to utilize under-sampling [28]. This technology uses the concept of aliasing to its advantage, rather than either

over-sampling a high frequency signal or mixing it down. By sampling at a pre-calculated rate, the receivers make the band-limited input signal alias itself down automatically onto a much lower frequency spectrum. Figure 2 shows a commonly used technique in cell phones and "software" radios. The architecture is separated into several sections to modularize the design.

For example, Global System for Mobile Communications (GSM) is the most popular standard for cell phones in the world today. Most GSM networks operate on frequency bands anywhere from 900 MHz to 1800 MHz [13]. Cell phones cannot use over-sampling in this case because a 4 GHz ADC is expensive, generates considerable heat, and quickly drains the battery life of the cell phone. Also, processing the data generated by a 4 GHz ADC requires a costly and complex digital architecture to handle all of the data. Instead, cell phones under-sample the frequency spectrum so that the frequency of interest aliases itself down onto a much lower frequency. This process allows for cell phones to use a much slower ADC which subsequently preserves power, lowers heat, reduces cost, and relaxes the computational requirements on the subsequent digital processing.

# III. Theory

This section discusses the theory of different methods of optical detection and the theory behind different methods of sampling.

## *Detection Theory*

There are two different methods that can be used to detect a light signal [8]. One scheme, known as *direct detection* and is a relatively simple method used in the majority of applications. The other scheme is known as *coherent detection* which requires a more complex optical architecture, but has certain advantages over direct detection. The key difference between these two methods is that with direct detection, the amplitude of the incoming signal determines the quantum-noise limit, whereas in coherent detection, the amplitude of the local oscillator signal determines the quantum-noise limit.

### Direct Detection

In a direct detection system, the signal beam of interest is incident directly unto the photo-detector which then generates a current to represent the detected beam of light. The complex electric field present upon the detector is represented as:

$$\hat{E} = Ee^{j(\omega t+\varphi)} \tag{1}$$

where $\omega$ = frequency (rad/s),
and $\varphi$ = phase shift (rad).

The current generated by the photo-detector can then be written as:

$$I_{ph} = \rho A \left\langle \left| \hat{E} \right|^2 \right\rangle \Big/ 2\eta_0 \tag{2}$$

where ρ = responsivity of detector,
A = receiving area of the detector,
$\eta_0$ = wave impedance in a vacuum,
and E = electric field.

Equation (2) shows that the photo-current is proportional to the average of the square of the magnitude of the electric field. The average should be over a time period much larger than the period of the optical wave, but less than the reciprocal of the electrical bandwidth of interest.

With direct detection, the quantum-limit signal-to-noise ratio (SNR) is:

$$S/N = \sqrt{\frac{I}{2eB}} \tag{3}$$

where e = charge of an electron,
and B = electrical bandwidth.

## Coherent Detection

Another method of detection is known as coherent detection. This method is slightly more complex because the signal is combined with a local oscillator at the detector. To analyze the results of this scheme, the electric field of both the signal and the local oscillator beam is represented by:

$$\hat{E} = Ee^{j(\omega t+\varphi)} \tag{4}$$

$$\hat{E}_0 = E_0 e^{j(\omega_0 t+\varphi_0)} \tag{5}$$

where $\hat{E}$ = electric field of the signal beam,
and $\hat{E}_0$ = electric field of the local oscillator.

The electric field in the photo-detector current, Equation (2), is replaced with $\hat{E} + \hat{E}_0$, and the photo-current equation can then be written as:

$$I_{ph} = \left(\frac{\rho A}{2\eta_0}\right)\left[\left\langle\left|\hat{E}\right|^2\right\rangle + \left\langle\left|\hat{E}_0\right|^2\right\rangle + 2\,\mathrm{Re}\left\langle E_0 E^*\right\rangle\right] \tag{6}$$

Which can be simplified and rewritten as:

$$I_{ph} = \left(\frac{\rho A}{2\eta_0}\right)\left\{E^2 + E_0^2 + 2EE_0\left\langle\cos\left[\left(\omega-\omega_0\right)\,t+\left(\varphi-\varphi_0\right)\,\right]\right\rangle\right\} \tag{7}$$

Then, using the following substitution:

$$\mu = \left\langle\cos\left[\left(\omega-\omega_0\right)\,t+\left(\varphi-\varphi_0\right)\,\right]\right\rangle \tag{8}$$

Equation (7) can be rewritten in terms of the photo-currents caused by each source.

$$I_{ph} = I + I_0 + 2\mu\sqrt{II_0} \tag{9}$$

where I = photo-current generated from the signal beam,
and $I_0$ = photo-current generated from the local oscillator.

Both of these terms appear as direct currents out of the photo-detector and the difference in frequencies between the two currents is represented by the rightmost term of Equation

(9). Thus, the amplitude of the beat frequency of the currents is directly proportional to the square root of the product of the two currents. This interference term is a key advantage of coherent detection over direct detection, because it achieves a gain in the detected signal. The gain is calculated by comparing the photo-currents in the coherent detection case versus the direct detection case.

$$G = \frac{\left[I + 2\mu\sqrt{II_0}\right]}{I} = 1 + 2\mu\sqrt{\frac{I_0}{I}} = 1 + 2\mu\frac{E_0}{E} \tag{10}$$

Equation (10) shows that by increasing the strength of the local oscillator, the strength of the detected signal of interest increases. This gain becomes important when trying to measure signals with small amplitudes, because the response of such signals can become dominated by noise. An ideal optical system is dominated by quantum-noise which allows it to achieve the highest signal-to-noise ratio (SNR) possible [8]. Thus, in a coherent system, the SNR can be improved by increasing the strength of the local oscillator beam to the point that the system becomes quantum-noise limited.

There are two sub-categories of coherent detection systems: *homodyne* and *heterodyne*. When the frequency of the local oscillator matches the frequency of the signal, the system is known as a homodyne system, otherwise, it is known as a heterodyne system. With homodyne detection, the signal and the noise is given by the same terms as seen in direct detection, plus the quantum-noise of the local oscillator photons. This fact presents a key disadvantage when working with low-amplitude signals. In heterodyne detection, the signal of interest is the beat frequency term caused by the difference in frequencies between the signal and local oscillator. Assuming the

current generated by the local oscillator meets the condition in Equation (11), the system becomes dominated by quantum noise.

$$I_0 >> I + I_d + I_R \tag{11}$$

where I = signal current,
$I_0$ = local oscillator current,
$I_d$ = photo-detector dark current,
and $I_R$ = terminating load current.

With homodyne detection, the quantum-limit SNR is:

$$S/N = \sqrt{\frac{4\mu^2 I}{2eB}} \tag{12}$$

where $\mu$ = cosine of the phase between the signal and local oscillator.

The SNR of a quantum-limited heterodyne system can be written as:

$$S/N = \sqrt{\frac{4\mu_\phi^2 I}{2eB}} \tag{13}$$

where $\mu_\phi$ = average of the cosine of the beat frequency between the signal and local oscillator.

## *Sampling Theory*

There are two different methodologies used in sampling continuous-time data and representing it in the discrete-time domain [11]. The most common method is known as over-sampling and it is categorized by sampling with a rate greater than twice the highest frequency component present in a signal. Under-sampling uses the property of aliasing to

sample a high-frequency signal at a rate below twice the highest signal frequency and have the signal alias to a lower frequency band in the discrete-time domain.

## Over-sampling

In traditional sampling systems, the minimum sampling rate should be greater than twice the highest frequency component in order for the signal to be properly reconstructed in the digital domain. The minimum sampling rate can be represented as:

$$f_S \geq 2 f_H \tag{14}$$

where $f_S$ = sampling rate,
and $f_H$ = highest input signal frequency.

Sampling systems that obey Equation (14) are referred to as over-sampling systems. In practice, the input to an over-sampling analog-to-digital converter is preceded by a low-pass filter in order to attenuate any higher frequency components and prevent them from aliasing onto the low frequency signal.

## Under-sampling

Another method for sampling an analog signal into the digital domain is known as under-sampling. There are a variety of different names for under-sampling such as: sub-sampling, direct IF-to-digital conversion, and harmonic sampling [16].

The Nyquist rate is defined as twice the bandwidth of a signal. The key concept behind under-sampling is that the Nyquist rate is not dependent on the highest frequency component in the signal, only the bandwidth of the signal. In order to properly represent the signal, it must be sampled at a rate above the Nyquist rate. Figure 3 shows an example frequency spectrum of a signal.

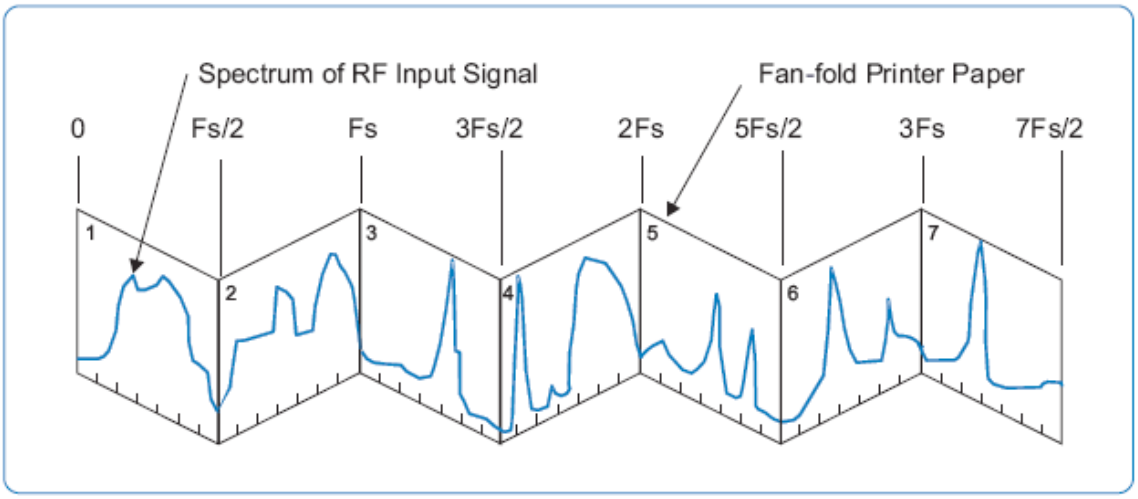



**Figure 3 - Example Frequency Spectrum [20]**

If the frequency spectrum above is sampled at a rate of $F_S$, then the spectrum is divided into regions every $F_S/2$ Hz. Figure 4 shows these regions then fold back onto each other (with every even numbered spectrum being reversed). This concept is known as aliasing, and once a signal is sampled, any aliasing cannot be removed.

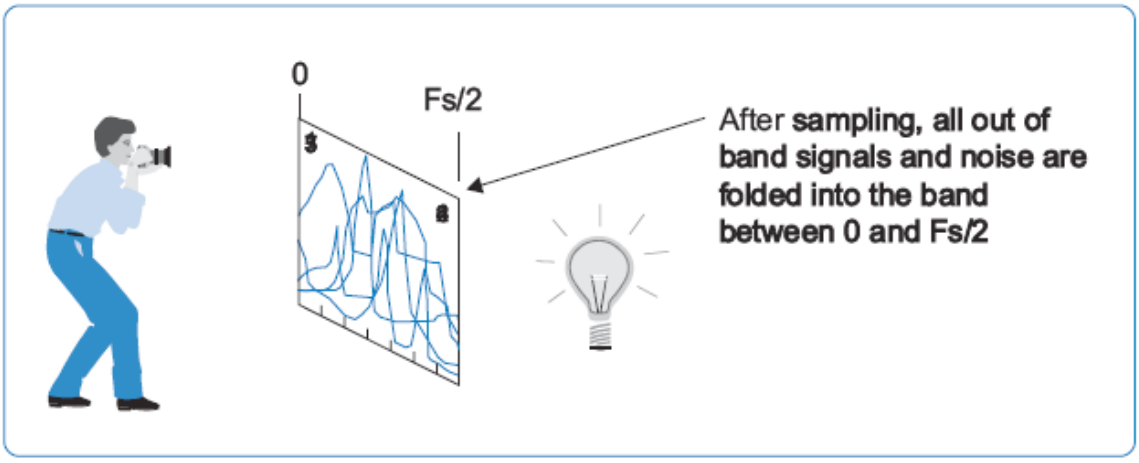



**Figure 4 - Aliased Frequency Spectrum [20]**

However, if the original signal of interest is band-limited, it can be sampled at a rate much lower than twice the highest frequency of interest, as long as the sampling rate is greater than twice the bandwidth of the signal. In this case, the property of aliasing becomes advantageous. Figure 5 shows an example of a band-limited spectrum.

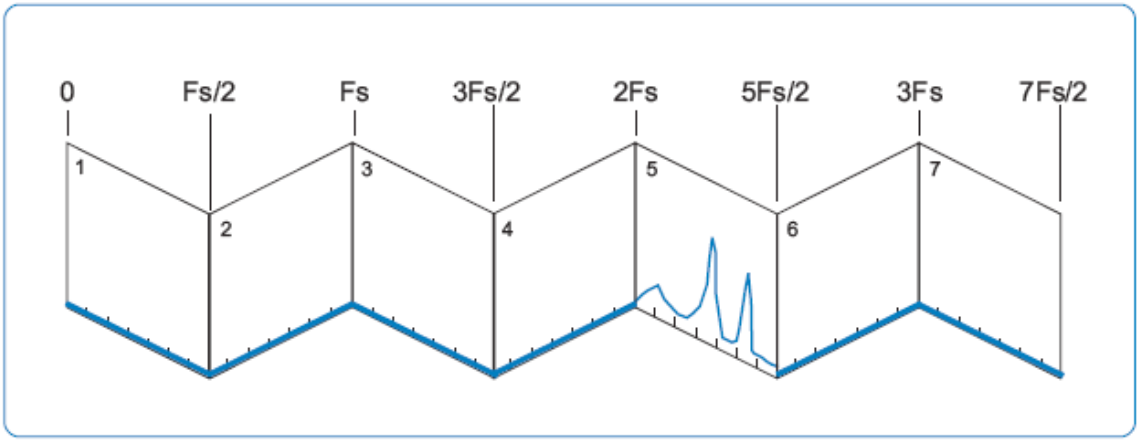



**Figure 5 - Band-Limited Frequency Spectrum [20]**

If the signal is sampled at a rate of $F_S$, all the regions again fold back onto each other. Fortunately, there are no frequency components in any of the other regions, so when the regions are folded together, the original frequency spectrum reappears as a baseband signal (Figure 6).

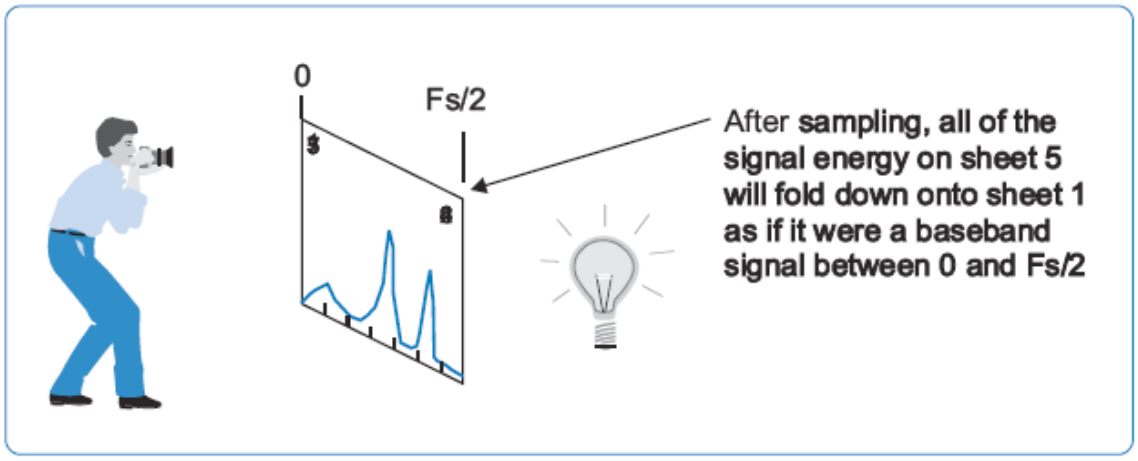



**Figure 6 - Aliased Band-Limited Frequency Spectrum [20]**

Choosing the sampling rate for under-sampling is not as straight forward as it might appear. As stated previously, the first requirement is that the sample rate must be at least twice the bandwidth of the signal. The second requirement is that the frequency regions do not intersect the spectrum of interest. If any of the $F_S/2$ borders fall in the middle of the spectrum of interest, part of the spectrum is aliased onto other parts of the signal, and the signal becomes corrupt.

# IV. Advantages of Multiple Channels and Under-Sampling

This section explains the benefits of multiple-channels and under-sampling in a laser vibrometry system.

## *Multiple Channels*

There are several benefits to using multiple channels in a laser vibrometry system. Normally, when a coherent laser is targeted on an optically rough surface, the reflected light is scattered into many different directions. Any photo-detector which attempts to measure the scattered light can only capture a small portion of this reflected light. Since such a small amount of energy is returned compared to the incident light, this process dramatically degrades the SNR. If the scatter is assumed to follow a Lambertian distribution, the power incident on the detector is calculated with Equation (15).

$$P_{\text{det}} = P_{laser} \frac{\rho A}{\pi L^2} \tag{15}$$

where $P_{Laser}$ = output optical power,
$\rho$ = reflectivity of the target surface,
A = receiving surface area on the photo-detector,
and L = distance from the target.

For example, with the given system conditions, an optical output power of one watt has an expected return optical power of approximately 100 nW. Because the return optical power is so low, it can become dominated by noise such as the relative intensity noise of the laser, thermal noise, etc. By having multiple detectors, more of the scattered return light is detectable, thus decreasing the effects of noise on the system.

Another major benefit of having multiple channels is that it allows for the effects of atmospheric turbulence to be dramatically decreased [14]. The scattered light that is reflected off a target must travel back through the atmosphere to reach the photo-detector. Since the atmosphere is not a perfect vacuum, small turbulence occurs. This turbulence is caused by random variations of pressure and temperature in the air which can slightly change the index of refraction at some points along the return path. These variations in the refractive index can cause additional random phase delays for some of the scattered light, which shows up as noise in the system. By having multiple receiving channels, the effects of the turbulence can be averaged out when data from multiple channels is combined together.

If the vibration measurement system is on any type of moving or vibrating platform, it causes added vibration noise to the detected signal. Because the vibrations detected by the system are on the order of a nanometer, even wind blowing against the system could cause additional vibration noise. With multiple-channels, this common-mode vibration noise should be present in all channels and can be filtered out with signal processing algorithms.

Another issue to consider is the structure of the target to be analyzed. If the structure is a building for example, different points on the outside of the building may vibrate differently depending on the internal structure of the building and the materials used in its construction. Moreover, vibrations may not occur in some locations due to this internal structure. By analyzing many individual points on the structure simultaneously, the probability of finding useful vibration data increases.

### *Under-sampling*

There are many advantages to using under-sampling over traditional over-sampling in a multiple-channel laser vibrometry system. One of the most important requirements in the system is to obtain the highest SNR possible so that the accuracy of the vibration measurements remains satisfactory. In a traditional over-sampling system, the beat frequency between the shifted reference beam and the incoming signal beam has to remain relatively small (around a few megahertz) [29]. Unfortunately, the relative intensity noise (RIN) from the laser can become an issue because the noise is strong at lower frequencies. Filtering out this RIN is a challenging task when the signal of interest is relatively close in frequency to the noise. If under-sampling is used, this beat frequency and thus, the signal of interest, can be pushed out to higher frequencies (i.e.: 40 MHz), where filtering out the RIN noise becomes much more straight forward. This technique allows for a much better SNR to be obtained, which is quantum-noise (also referred to as shot-noise) limited if the system is designed properly.

As the term under-sampling implies, the data is sampled at a much lower rate than in an over-sampled system. This process results in a much lower frequency sampling system being required. Since many integrated circuits are based on processes such as CMOS whose power requirements increase linearly with frequency, a low frequency clock requires much less power. The result is that the requirements on the system power supply are significantly reduced. Since less power is being consumed, less heat is dissipated by the components, which relaxes the cooling requirements of the system. Moreover, when the heat is reduced, the components can be placed closer together in the system, allowing for a more compact form-factor.

Another advantage to under-sampling is that only one acoustic-optic modulator (AOM) is required in the system instead of two to generate the reference beam. AOMs allow light that is passed through them to be shifted by a given RF input frequency. The available frequency shift range for most AOMs is from 40-60 MHz. The photo-detector would then generate a carrier frequency equal to the AOM frequency shift and thus require a high-speed ADC. One way to reduce the required sampling speed of the ADC in an over-sampling system is to use two AOMs in series, each with a slightly different frequency shift. For example, a 55 MHz up-shifting AOM in series with a 51 MHz down-shifting AOM produces a net frequency shift of only 4 MHz. Since the sampling rate of the under-sampling system is not dependent upon the carrier frequency, a higher optical frequency shift is acceptable, and only a single AOM is required. This component reduction results in cost savings, decreased heat generation, and a smaller system footprint.

A third advantage to using under-sampling is that once data is sampled into the digital domain at a given rate, it must be initially processed at that rate. When dealing with multiple individual channels, processing the data coming out of all the ADCs becomes a challenging task. The over-sampling system samples at a higher rate than the under-sampling system, and thus more digital data is generated by the ADCs with over-sampling. All the resultant data must be processed either with FPGAs, ASICs, or microprocessors. More data requires more gates on the FPGA, more silicon in the ASICs, or more complex microprocessors, all of which translate to increased cost, power, and heat generation.

# V. Practical Considerations

The concept of under-sampling is simplistic in theory; however, there are a number of practical issues that arise when actually designing an under-sampling system. These issues are either not present or much less significant when working with classical over-sampling systems. These issues include jitter, ADC analog bandwidth, and aliasing with digital down-conversion.

One of the biggest issues with an under-sampling system is the effect of sampling jitter [15]. Jitter is the variation in period from cycle to cycle of the clock signal. Because the clock signal controls the sampling times of the ADC, any jitter present on the clock, appears as noise in the sampled data stream. The amount of noise generated by the jitter is dependent upon the jitter time and the rate of change of the incoming signal. With a fast-changing signal, sampling at the incorrect time can cause larger deviations in the measured data as opposed to the same sampling of a slow-changing signal. Figure 7 illustrates the effect of jitter in a traditional over-sampling system.

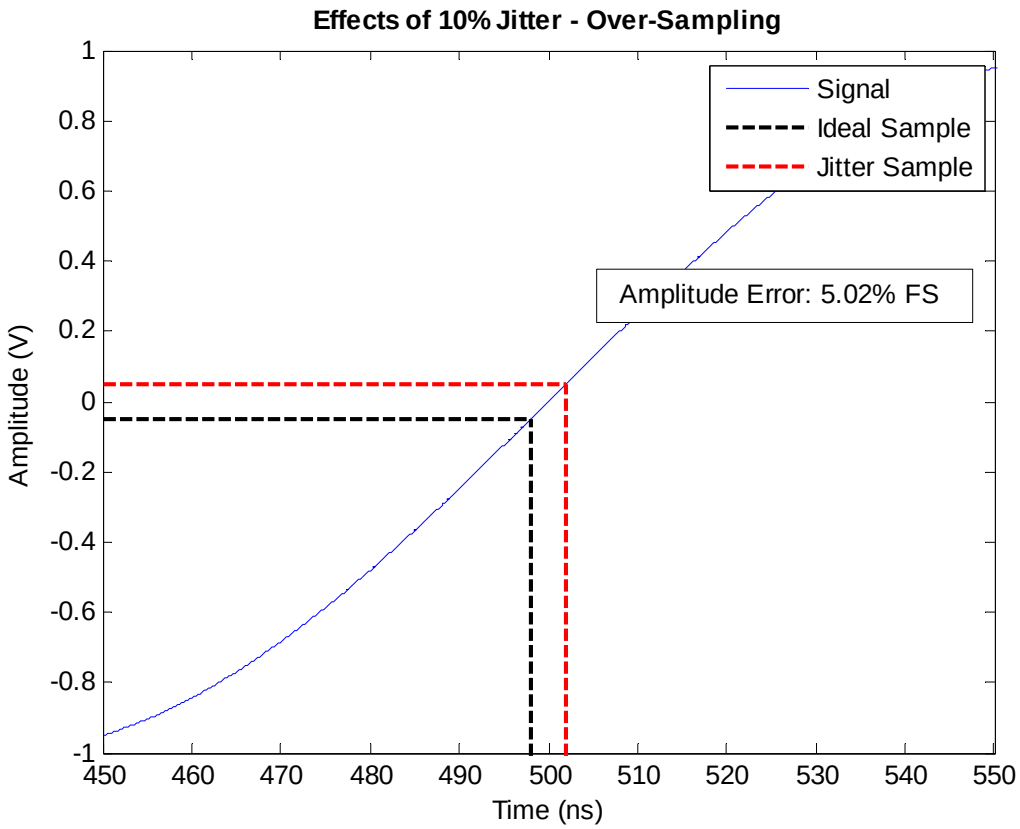



**Figure 7 - Over-Sampling Jitter Diagram**

In an under-sampling system, the negative effects caused by jitter are greatly magnified. The sample point on an over-sampling system is shifted only slightly in comparison to the length of a full sinusoidal cycle. However, in an under-sampling system, a sample point may only be gathered once every several cycles or less. A small sample-time deviation in this case places the sample point multiple cycles away from the intended time location. Figure 8 shows the magnified effects of jitter in an under-sampling system.

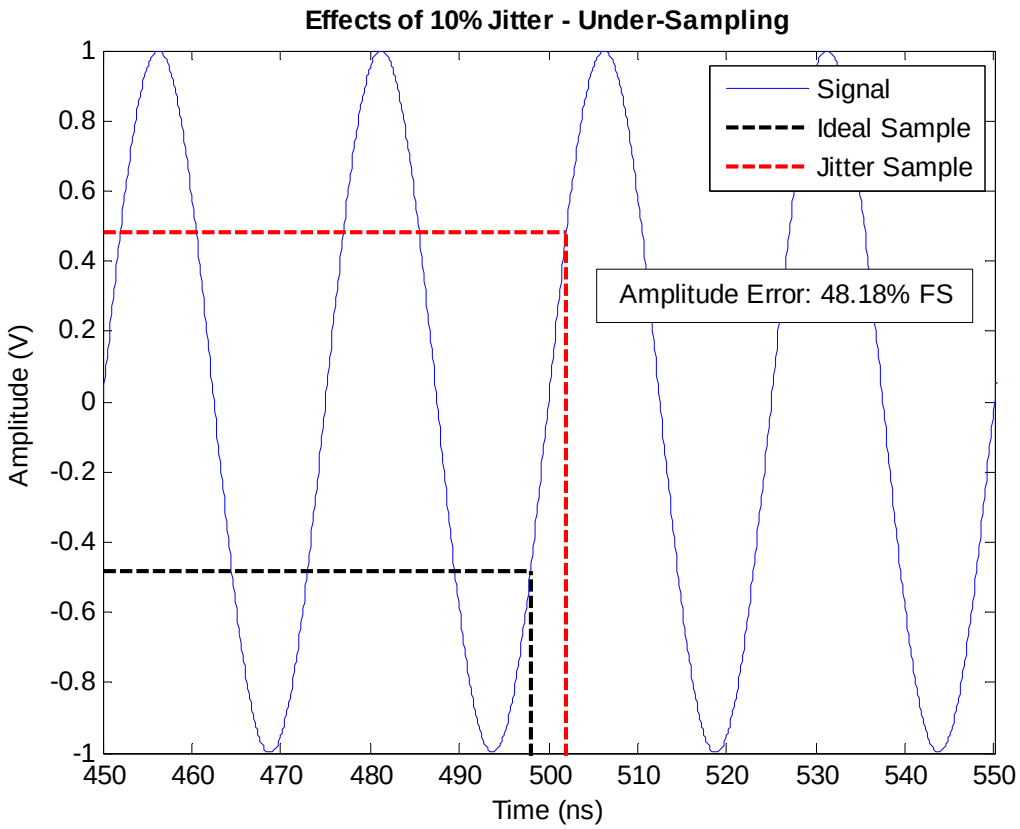



**Figure 8 - Under-Sampling Jitter Diagram**

In a laser vibrometry system, where the goal is to measure phase changes of only several degrees, the jitter requirements are much stricter. The amount of jitter places an upper limit on the achievable system SNR and can be calculated with Equation (16).

$$SNR_{Jitter} = 20 \times \log\left(\frac{1}{2\pi \times f_{in} \times t_j}\right) \tag{16}$$

where $f_{in}$ = input signal frequency,
and $t_j$ = root-sum square of all jitter sources.

These jitter sources include clock jitter, analog input signal jitter, and ADC aperture jitter. As an example, if the input frequency is 40 MHz and there is 5 ps of RMS jitter, the maximum obtainable SNR from Equation (16) is:

$$SNR_{Jitter} = 20 \times \log\left(\frac{1}{2\pi \times 40e6 \times 5e-12}\right) = 58.016dB \quad (17)$$

Another consideration when designing an under-sampling system is selecting a high-quality ADC. In addition to the standard set of considerations given to an ADC for an over-sampling process, the analog input bandwidth of the ADC also becomes a concern. For example, when under-sampling, the sample rate may be only 12.5 MHz, but the incoming analog signal frequency might be 40 MHz. This requires that the analog input portion of the ADC has to be able to accommodate the higher frequency signal. The requirements on the sample-and-hold amplifier within the ADC are more stringent with regard to the sample time. Most ADC datasheets provide the analog input bandwidth specification, and list whether or not the ADC is suitable for under-sampling applications.

Aliasing with digital down-conversion must also be taken into account when designing the under-sampling system. Besides placing an analog band-pass anti-aliasing filter ahead of the ADC input, the sampling rate must be chosen carefully or aliasing in the digital domain may occur. For example, if a system with an analog input frequency of 40 MHz is sampled at a rate of 12 MHz, the signal is aliased down to 4 MHz. Digital-down conversion (DDC) is then used to digitally mix the 4 MHz signal down to baseband. DDC involves multiplying the signal by a 4 MHz sinusoid in the digital domain, which is equivalent to mixing the signal in the analog domain. From this mixing

process, two frequency-shifted signals are generated, one at the sum of the two mixed frequencies and one at the difference of the two mixed frequencies. The resulting signal contains both a DC component and an 8 MHz component. Since a sample rate of 12 MHz is being used, the first Nyquist region ranges from DC to 6 MHz, and the 8 MHz component folds back into the first region, reappearing at 4 MHz. Figure 9 illustrates the digital aliasing example. The amplitude of the aliased signal is equal to the amplitude of the baseband signal of interest, so the closer in frequency it is to the baseband signal, the more difficult it becomes to fully attenuate the unwanted component. The result is additional noise being added onto the signal.

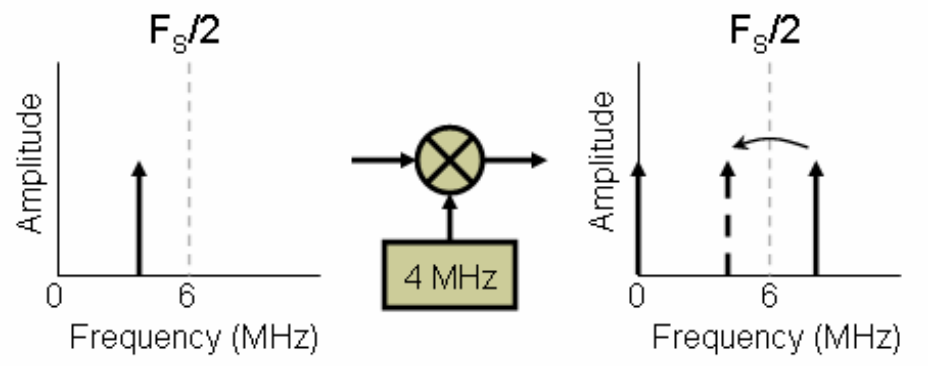



**Figure 9 - Digital Aliasing Diagram**

# VI. System Overview

This section describes the optical and electrical architecture used in the vibrometry system. The primary objective for the vibrometry system is to measure motion signatures on buildings' exterior walls from mechanical sources within. The amplitude of these vibrations is normally on the order of a nanometer and their frequencies of interest are mostly within the audible range, although they could possibly range higher. The system also needs to work at a significant distance from the target of interest. The optical and electrical overview given below is for a single-channel system.

## *Optical Flow*

A laser source generates a coherent beam of light. The beam of light is then split into a "signal" beam and a "reference" beam. The "signal" beam hits the target of interest, and any vibrations from the target are phase-modulated onto the "signal" beam. The phase-modulation of the signal beam can be represented as:

$$\cos\left[\omega_s t + 2\pi \frac{2A_{Vib}}{\lambda_S}\cos\left(\omega_{Vib} t\right)\right] \tag{18}$$

where $\omega_S$ = optical angular frequency,
$\omega_{Vib}$ = angular frequency of the target vibration,
$A_{Vib}$ = target vibration amplitude,
and $\lambda_S$ = optical beam wavelength.

The "reference" beam is passed through an acousto-optic modulator (Bragg Cell) which shifts the frequency of the "reference" beam [8]. Both the "reference" and the "signal" beam are then recombined and sent into a photo-detector. This detection method is

known as a heterodyne or coherent detection. The power incident on the detector can be written as:

$$P_{\text{det}} \propto \left| E_S + E_{\text{Re}\,f} \right|^2 \tag{19}$$

where $E_S$ = signal beam electric field,
and $E_{Ref}$ = reference beam electric field.

The current output from the detector can be written as:

$$i_{\text{det}} = \eta_h \eta_c \rho_{Det} \left( P_{\text{Re}\,f} + P_S + \sqrt{P_{\text{Re}\,f} P_S} \cos\left[ \left( \omega_{ref} - \omega_S \right) t + 2\pi \frac{2A_{Vib}}{\lambda_S} \cos\left( \omega_{vib} t \right) \right] \right) \tag{20}$$

where $\rho_{Det}$ = photo-detector responsivity,
$P_{Ref}$ = reference beam power,
and $P_S$ = signal beam power.

Figure 10 shows a block diagram overview of the optical system and the major components associated with it.

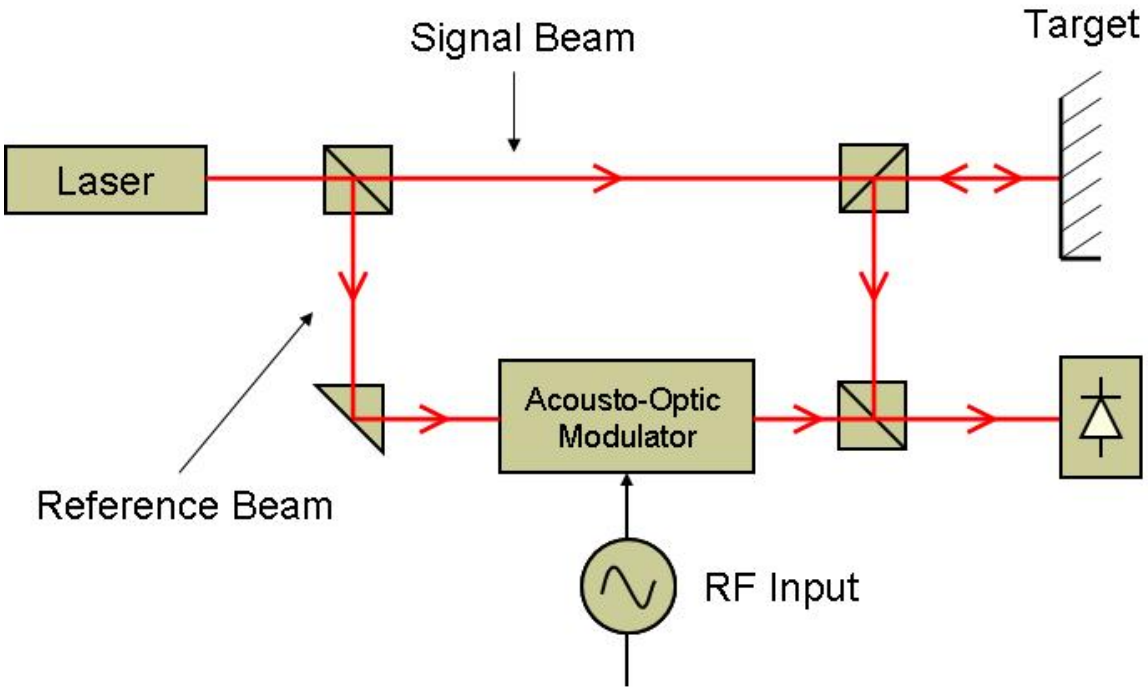



**Figure 10 - Optical System Diagram**

## *Electrical Flow*

Figure 11 shows a block diagram overview of the electrical flow.

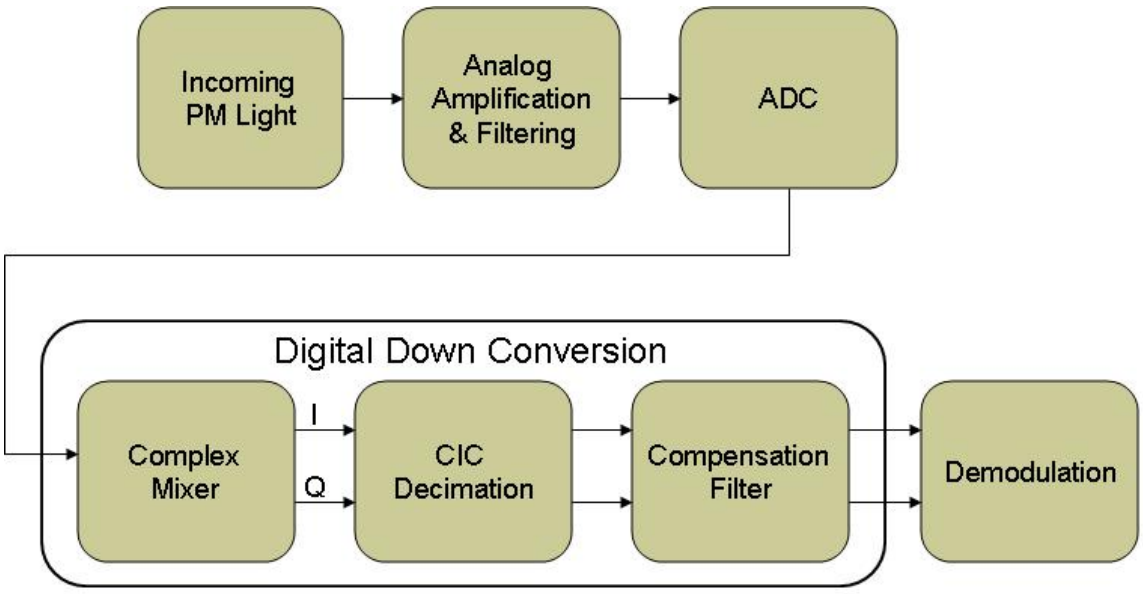



**Figure 11 - Electrical System Flow Diagram**

From an electronics perspective, the incoming light on the photo-detector is phase modulated at a carrier frequency of the beat frequency between the reference and the signal beam. There are also DC current components present at the photo-detector from the optical power of the two beams. The photo-detector feeds the AC-coupled input of a trans-impedance amplifier (TIA) to convert the current to a voltage signal for further processing. The TIA is a low-noise amplifier with a high gain of approximately twenty thousand. From the output of the trans-impedance amplifier, the signal is now a fully-differential signal to minimize the effects of external noise. The signal passes through a narrow LC band-pass filter into another amplification stage with a gain of five. The signal then goes through an identical band-pass filter and amplification stage again. Afterwards, the signal is passed through a high-pass filter to further attenuate any low-frequency noise and then amplified with a gain of four. The effect of these filtering and amplification stages is to provide a total gain of approximately two million while acting as a high-Q anti-aliasing filter for the input of the ADC. Figure 12 shows a diagram of the analog signal flow.

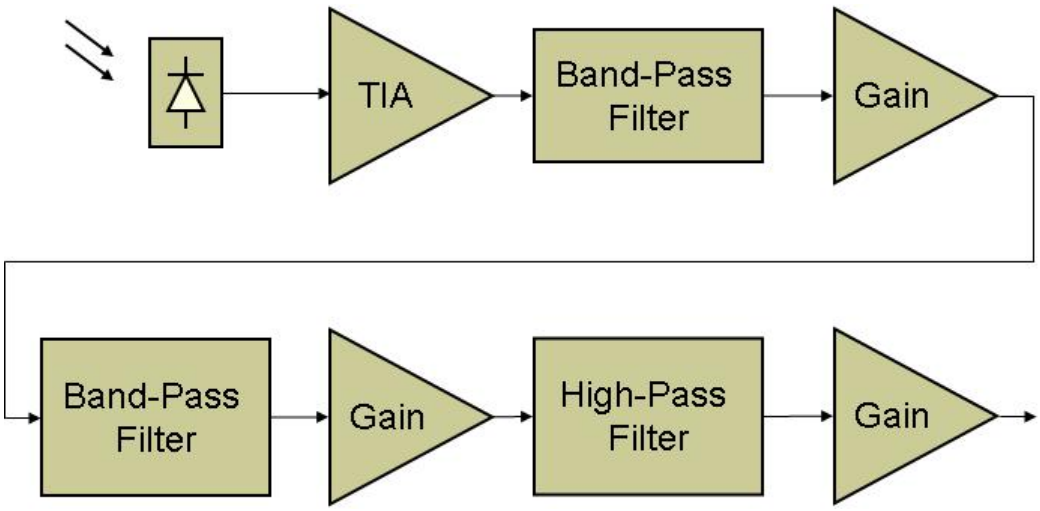



**Figure 12 - Analog Signal Flow**

The signal is then under-sampled by a low-jitter ADC, where it is passed to an FPGA for digital processing. The signal must first be digitally down-converted which is accomplished by mixing the signal with a numerically-controlled oscillator to generate two sinusoids (one shifted by 90°). This action produces two data streams from the original signal, an I and Q stream, known as quadrature modulation. Having two data streams is useful for demodulation of the signal because it allows for the amplitude dependence to be removed. Figure 13 shows the digital signal flow.

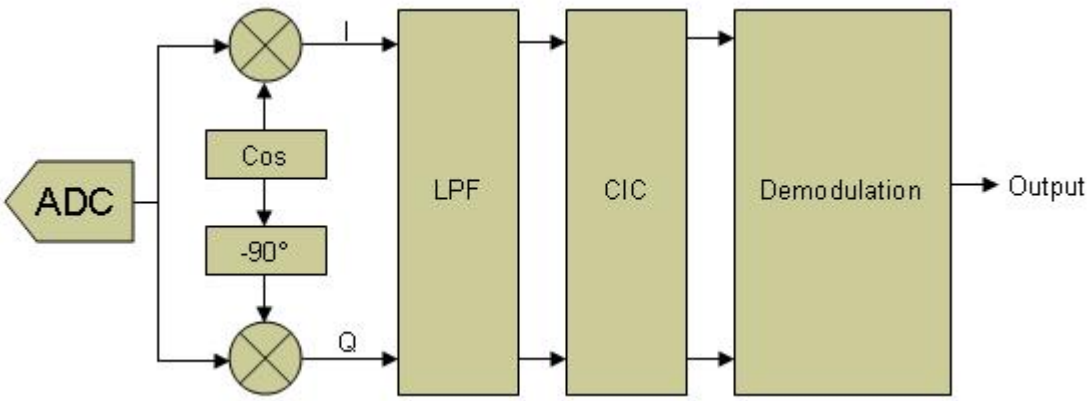



**Figure 13 - Digital Signal Flow**

After the I and Q streams are generated, the signals pass through a low-pass finite-impulse-response (FIR) filter to attenuate the unwanted higher-frequency components. At this point, the sample rate for the data is still 12.5 MHz. But since the frequencies of interest range from DC to 20 kHz, the data can be decimated to a much lower rate with the use of a cascaded integrator-comb (CIC) filter. This type of filter is known as a multi-rate filter, because it changes the sampling rate of the data from the input to the output. A CIC filter is made of two basic components: a single-pole IIR filter which acts as an integrator, and a comb filter which changes the sampling rate of the data. The integrator acts as a low-pass filter with unity gain and is described with Equation (21). Figure 14 shows the block diagram for this integrator, which consists of only an adder and a delay.

$$y[n] = y[n-1] + x[n] \tag{21}$$

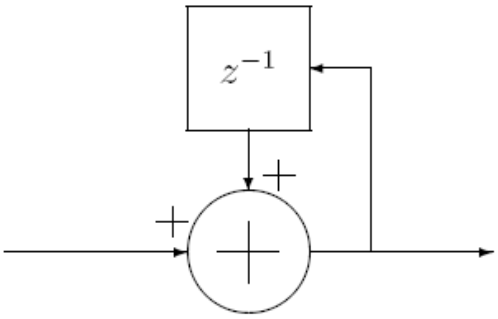



**Figure 14 - Block Diagram of a Single-Pole Integrator [7]**

The comb filter acts as an odd-symmetric FIR filter and is described as:

$$y[n] = x[n] - x[n - RM] \tag{22}$$

where R = rate change factor,
and M = differential delay.

The output sampling rate is then described with Equation (23).

$$f_{S-Out} = \frac{f_{S-In}}{R} \tag{23}$$

Figure 15 shows the block diagram of the comb filter, which consists of only a subtraction unit and a delay.

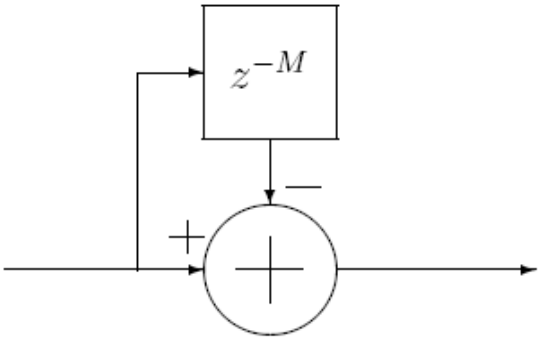



**Figure 15 - Block Diagram of a Comb Filter [7]**

These two basic building blocks can be chained together to achieve different levels of decimation. For example, Figure 16 shows a three-stage decimating CIC filter that is built by connecting the integration and comb filters together with a rate change of R.

.

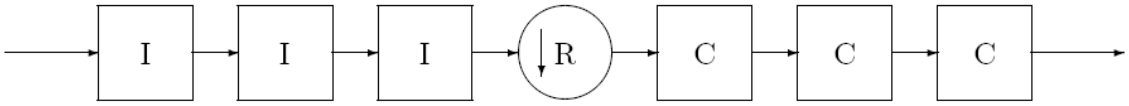



**Figure 16 - Three-Stage Decimating CIC Filter [7]**

After the data streams have passed through the CIC filter, their sample rates have been significantly reduced. For example, a signal sampled at 12.5 MHz may be decimated by a factor of 64 to result in the equivalent of a signal sampled at 195 kHz. The outcome is a first Nyquist region that ranges DC to 97.5 kHz. This reduced data rate allows for downstream filters to process data with a significantly lower number of taps. Moreover, the demodulation process requires much less computing power.

Once the sampling rate is decimated, the signal is processed to demodulate the phase-modulated information from the signal. The I and Q data streams from each channel are represented by the Equations (24) and (25).

$$I = \cos(\omega_{ref} t)\cos\left[\omega_{ref} t + 2\pi \frac{2A_{Vib}}{\lambda_S}\cos(\omega_{Vib} t)\right] \tag{24}$$

$$Q = \cos\left(\omega_{ref} t - \frac{\pi}{2}\right)\cos\left[\omega_{ref} t + 2\pi \frac{2A_{Vib}}{\lambda_S}\cos(\omega_{Vib} t)\right] \tag{25}$$

Equation (26) shows how the phase-modulated information from Equations (24) and (25) are then extracted by calculating the inverse tangent of the Q stream divided by the I stream. This demodulation technique allows the amplitude variations in the incoming light signal due to scattering and atmospheric turbulence to be removed from the data stream.

$$\phi = \arctan\left(\frac{Q}{I}\right) = 2\pi \frac{2A_{Vib}}{\lambda_S}\cos(\omega_{Vib} t) \tag{26}$$

# VII. Modeling and Simulation

In order to verify the theory before implementing the design, we created models and simulations to verify both the behavior of the system and the dominant noise sources.

## *System Modeling and Behavioral Simulation*

Simulink, which is a graphical block diagram and modeling tool, was used to design a functional model of the system. The electrical flow was broken down into sub-components and then each sub-component was modeled. The entire system was subsequently linked together to verify the operation. Figure 17 shows an overview of the entire model and all of its functional modules.

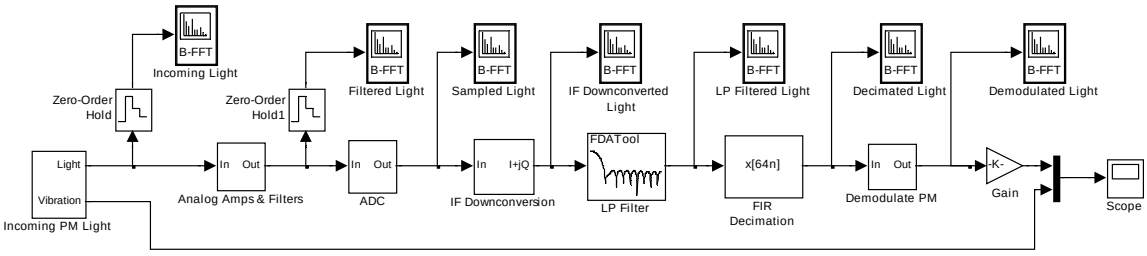



**Figure 17 - Diagram of Simulink Model**

First, an incoming phase-modulated light beam was modeled as the input seen by the photo-detector. Figure 18 shows how the light represented mathematically by Equation (18) is decomposed into smaller components and constructed in Simulink.

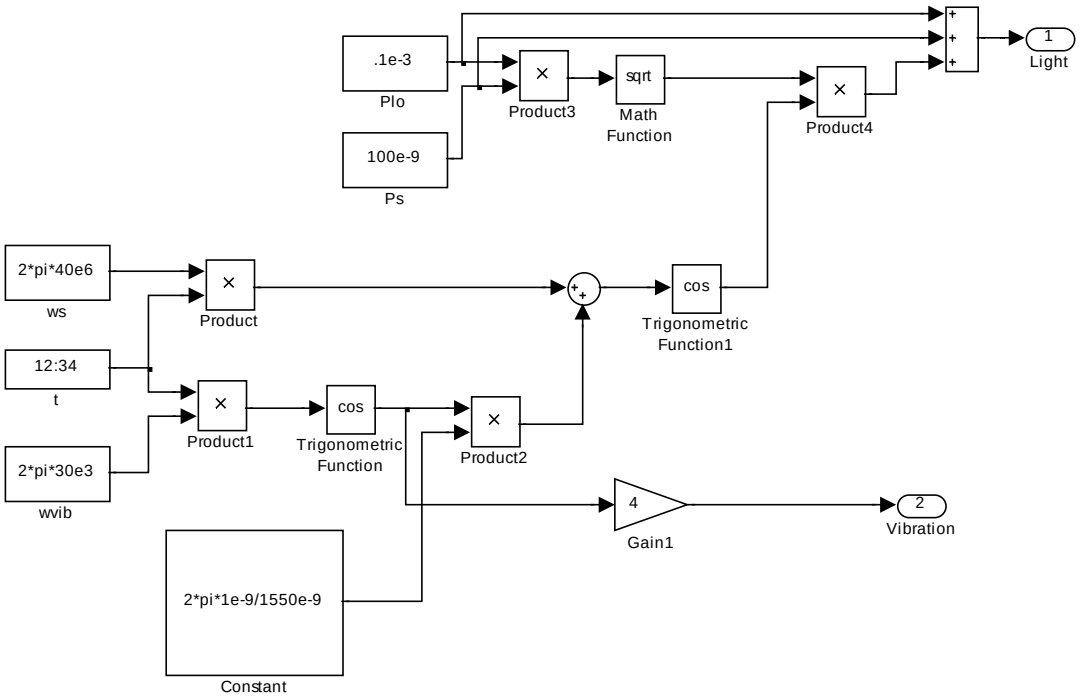



**Figure 18 - Model of Incoming Phase-Modulated Light**

As an example, an input vibration frequency of 30 kHz is used in this simulation which is seen in the $\omega_{vib}$ constant block of Figure 18. Figure 19 shows the spectrum of the incoming phase-modulated light, which resembles an impulse at the carrier frequency of 40 MHz.

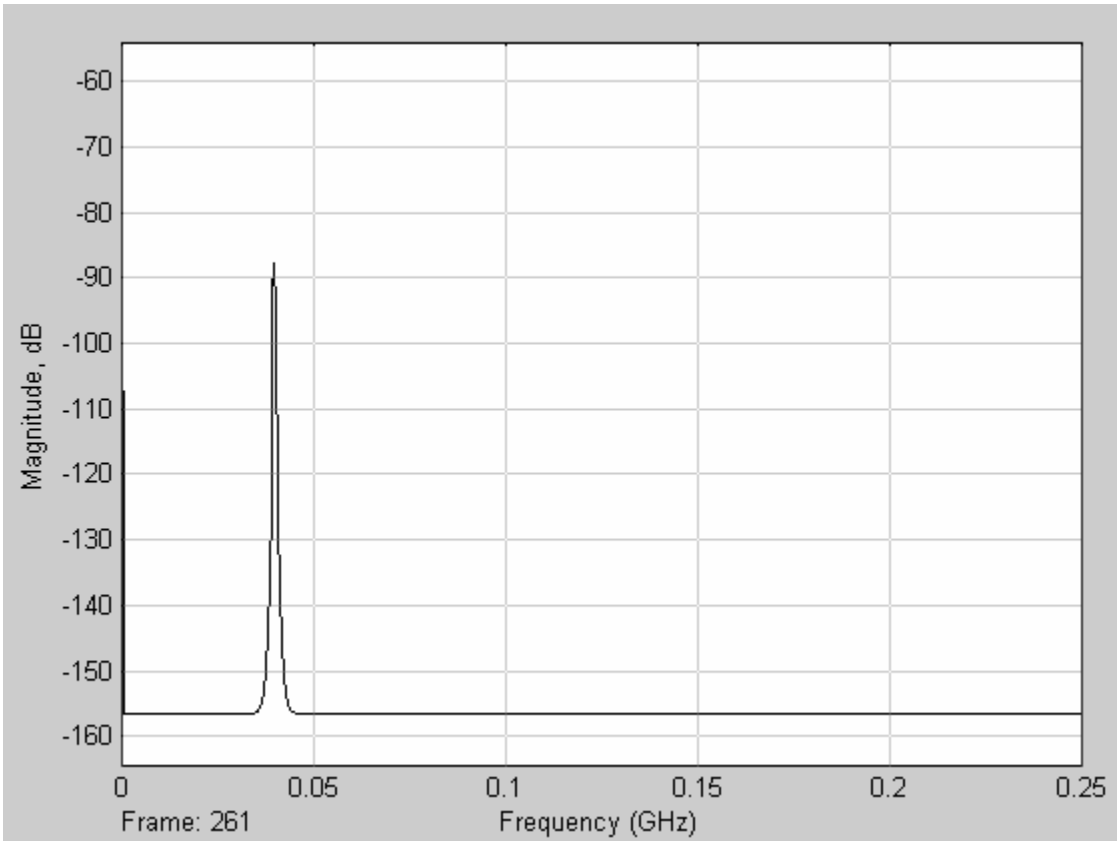



**Figure 19 - Simulated Spectrum of Simulated Incoming Light**

In Figure 19, the beat frequency from the reference beam mixing with the signal beam is at 40 MHz. The DC components from the light beams are also present.

Once the light is detected by the photo-detectors, it passes through a series of amplification and filtering stages. A sixth-order low-pass Butterworth filter is used to approximate the transfer characteristics of the multi-stage filter that is implemented in the real system. Figure 20 shows the analog amplification and filtering block and models the expected photo-detector responsivity as well as the gains of the trans-impedance amplifier and the subsequent voltage amplifiers.

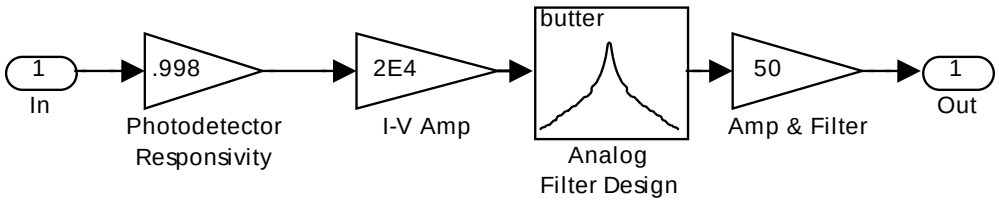



**Figure 20 - Model of Analog Amplification and Filtering**

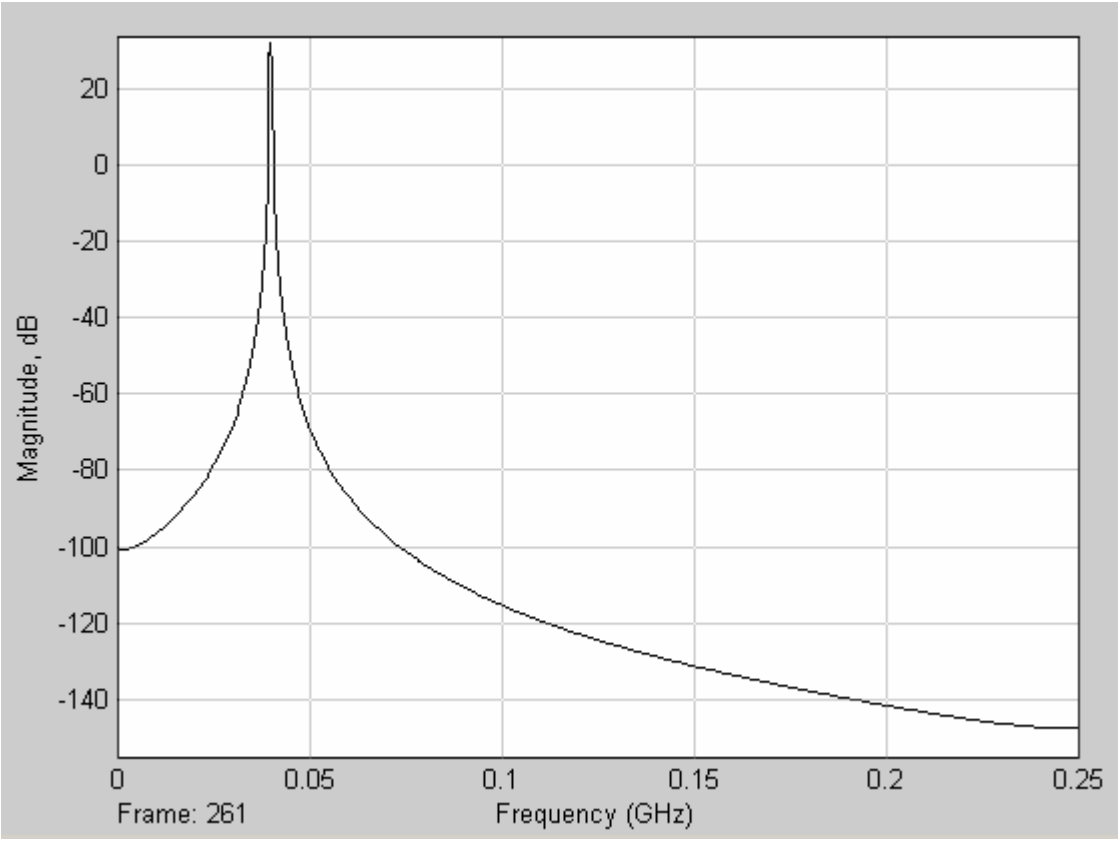



**Figure 21 - Simulated Spectrum of Analog Amplified and Filtered Signal**

Figure 21 shows the spectral output of this stage which is still an impulse at 40 MHz. In the output spectrum, the DC components are significantly attenuated and the frequency of interest, centered around 40 MHz, is amplified.

After amplification and filtering, the signal is sampled by an ADC. The ADC model takes into account the effects of both quantization and jitter to provide a more realistic simulation output. Figure 22 shows the block diagram of the ADC model.

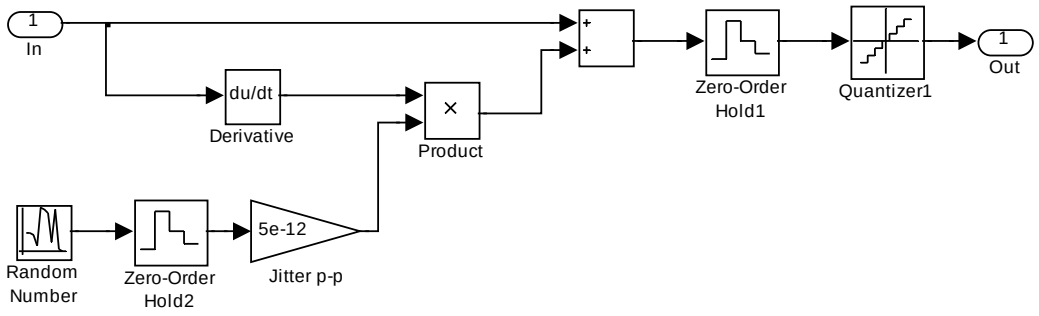



**Figure 22 - Model of ADC**

Figure 23 shows the spectral output after under-sampling. At this stage, the under-sampled 40 MHz signal appears at 2.5 MHz, which is the expected aliased frequency. The frequency axis in Figure 23 ranges from DC to $F_S/2$ (6.25 MHz), which is the first Nyquist region.

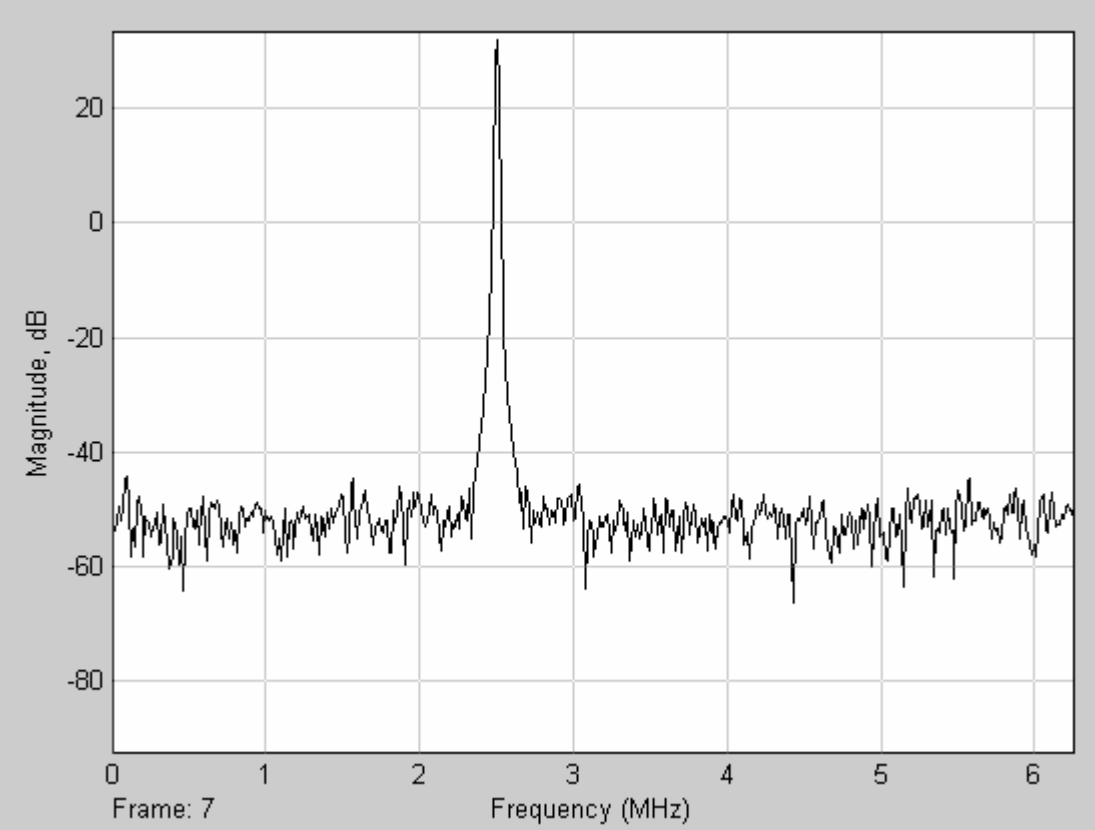



**Figure 23 - Simulated Spectrum of Under-Sampled Signal**

Once sampled, the signal is digitally-down converted to move the 2.5 MHz carrier frequency down to a base-band level. The I and Q data streams are generated at this stage of the process. Figure 24 shows a block diagram of this model.

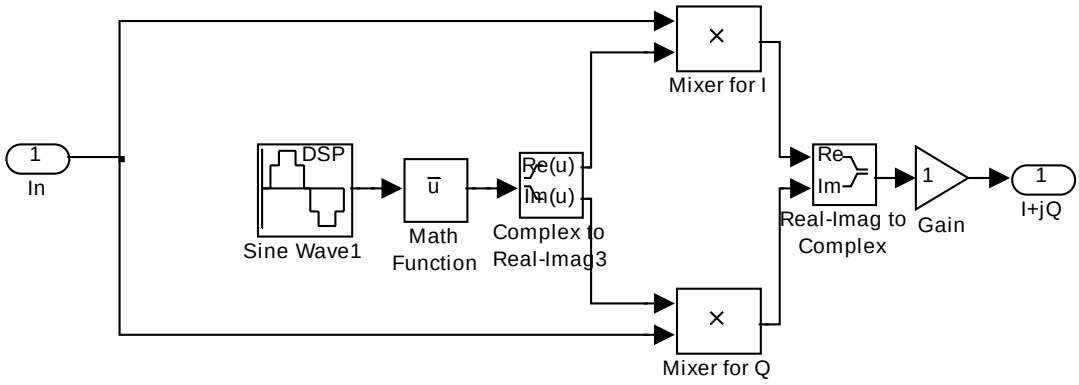



**Figure 24 - Model of Digital-Down Conversion [4]**

The numerically-controlled oscillator generates both sine and cosine waveforms which are represented as a complex stream of data. Each of the streams is then mixed with the sampled input to produce the I and Q channels. Figure 25 shows the spectrum of the output after down-conversion.

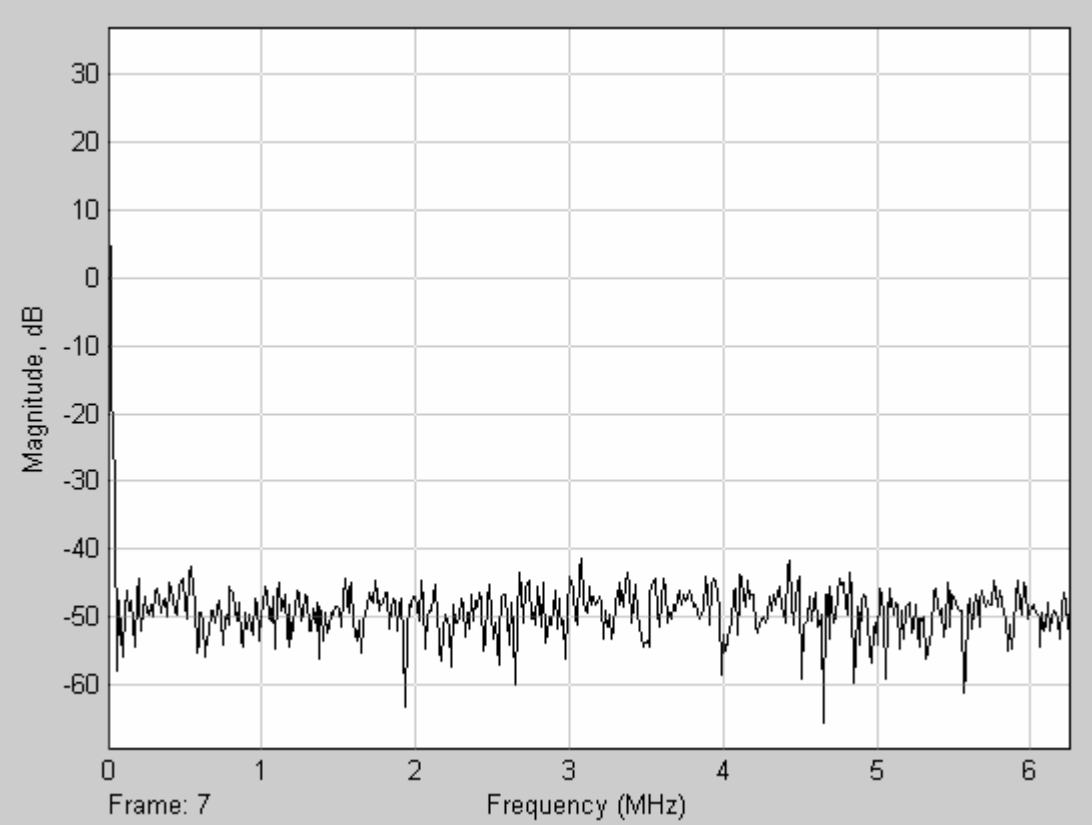



**Figure 25 - Simulated Spectrum of Down-Converted Signal**

The higher-frequency carrier is removed from the phase-modulated information at this point and the lower-frequency carrier has been shifted down to DC.

Once the signal is down-converted, it is low-pass filtered to attenuate any unwanted signals outside the frequency range of interest (0-50 KHz). This operation is accomplished with a low-pass FIR filter. The filter used during simulation was a $20^{th}$ order low-pass filter with a Hamming window and a cutoff frequency of 50 kHz. This filter was custom designed with the Filter Design and Analysis tool in MATLAB. Figure 26 shows the magnitude response of this filter.

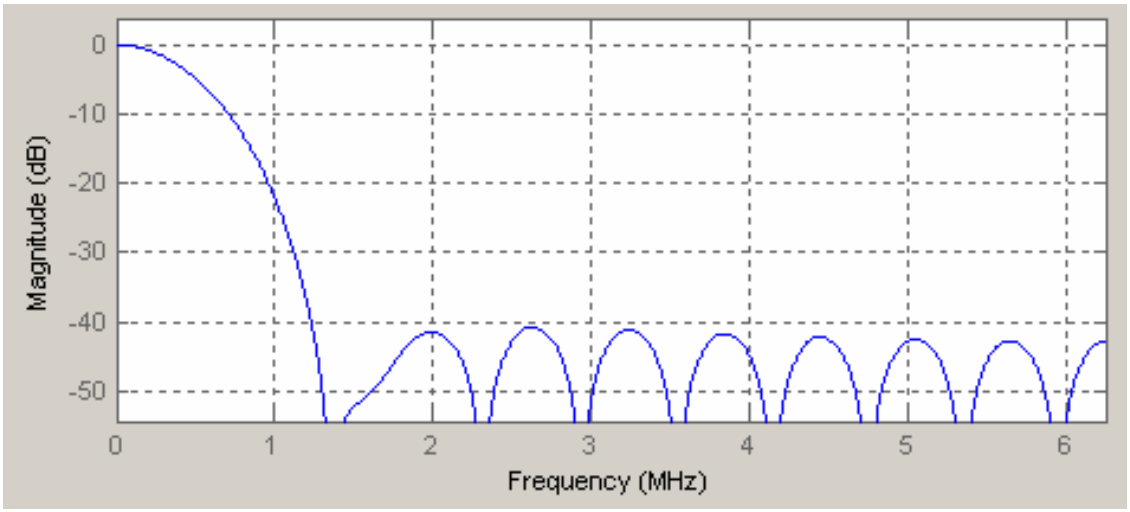



**Figure 26 - Simulated Magnitude Response of Low-Pass Filter**

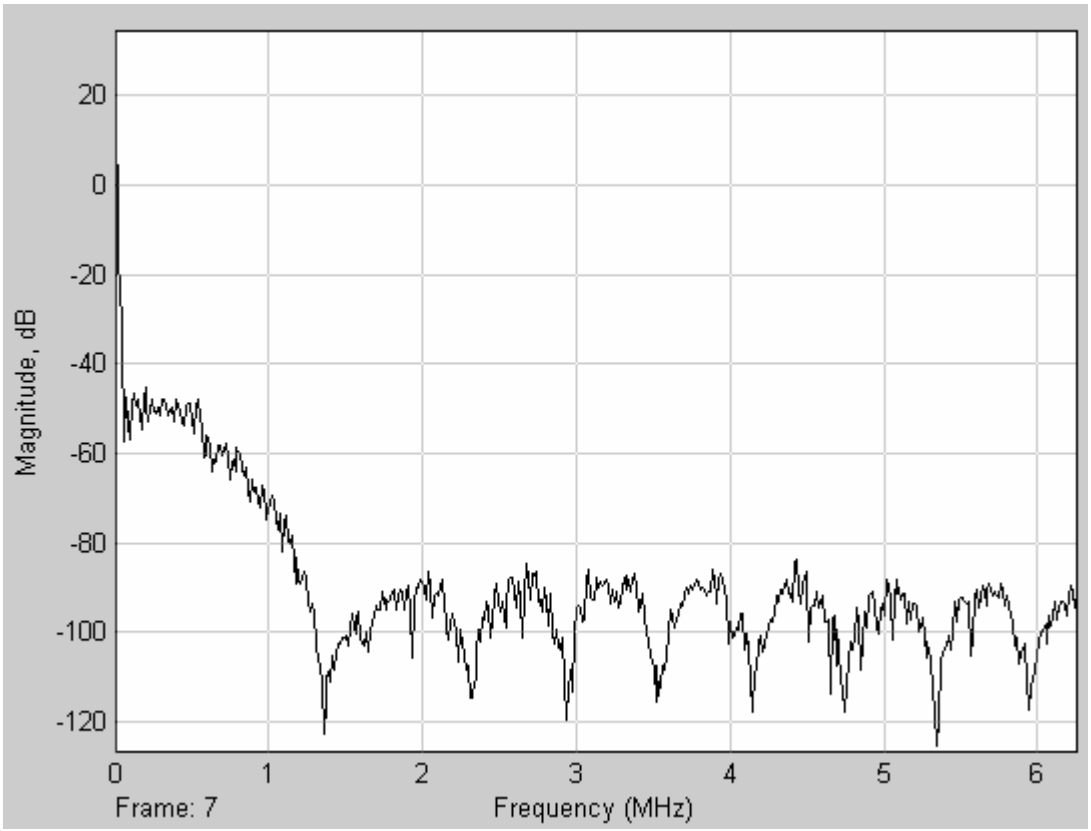



**Figure 27 - Simulated Spectrum of Low-Pass Filtered Signal**

Figure 27 shows the spectrum of the signal after passing through the low-pass filter. Once the signal is filtered, it is decimated in order to reduce the downstream processing requirements. In the simulation, this result was accomplished with an FIR decimation filter instead of a CIC decimation filter. The FIR filter was chosen over the CIC because the CIC filter model was not available for simulation and the transfer characteristics are similar. The FIR filter reduced the input sampling rate by a factor of 64, which creates an output sampling rate of approximately 195.3 KHz. Figure 28 shows the output spectrum of the decimation filter.

.

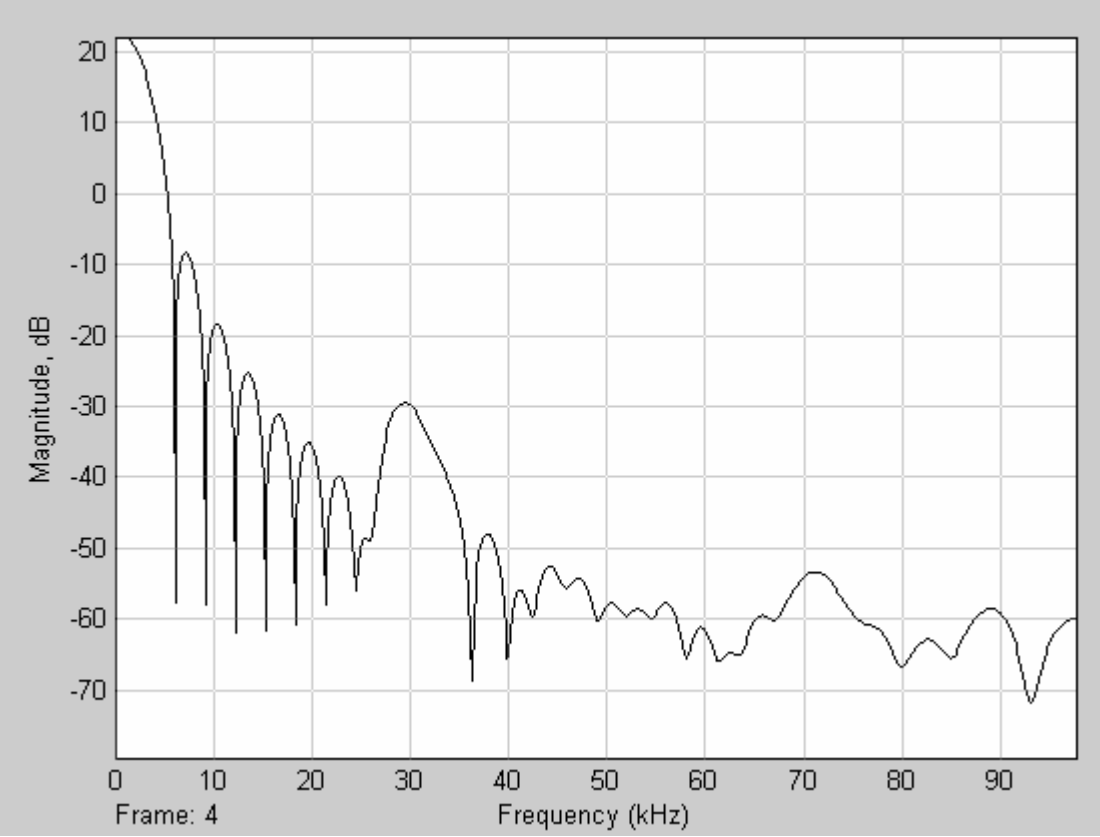



**Figure 28 - Simulated Spectrum of Decimated Signal**

After decimation reduces the sampling rate, the signal is demodulated to extract the original vibration information. Figure 29 shows the model of the demodulation block. This module divides the I stream by the Q stream and then calculates the inverse tangent of the result. Figure 30 shows the spectrum of the signal after demodulation. The spectrum contains a peak centered at 30 kHz, which is the original vibration frequency of the incoming light.

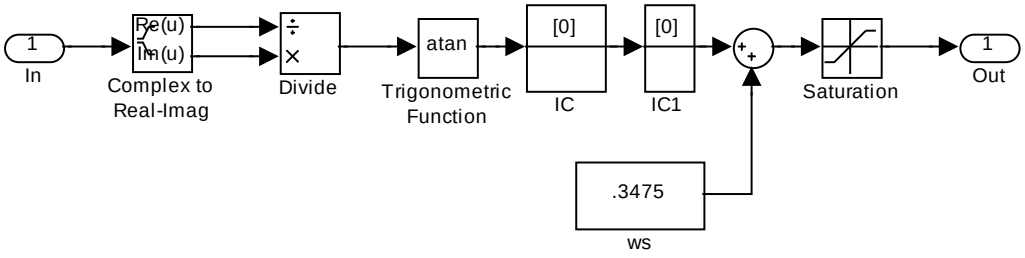



**Figure 29 - Model of Demodulation**

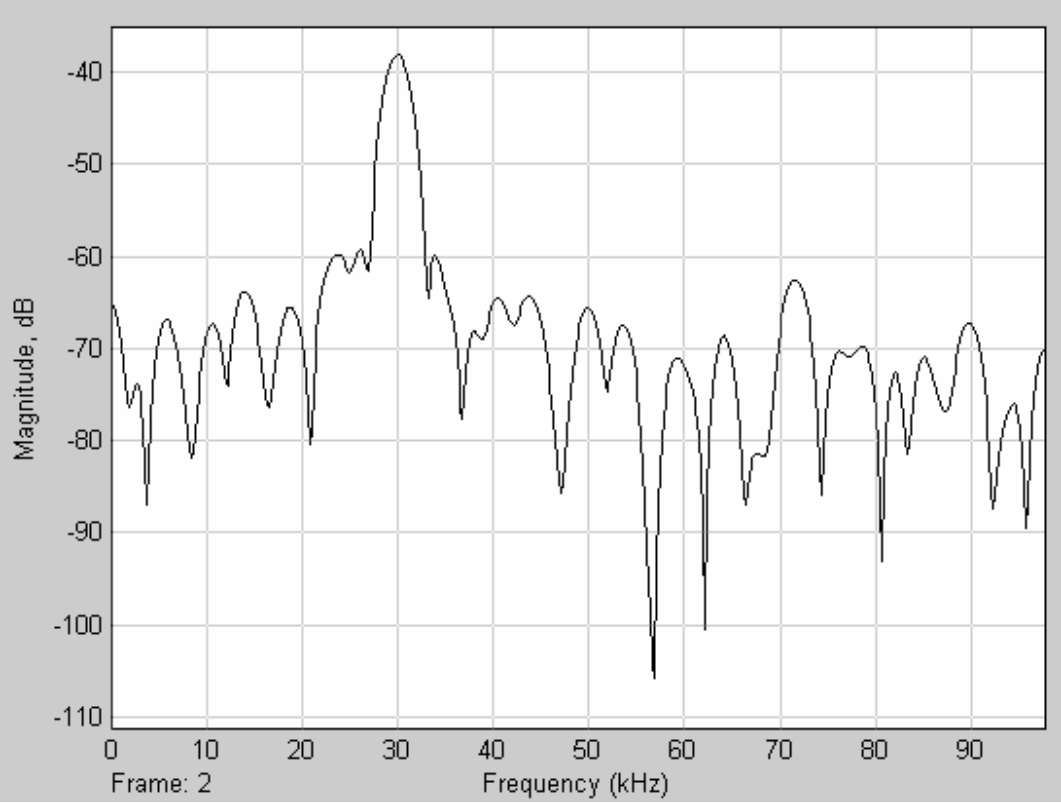



**Figure 30 - Simulated Spectrum of Demodulated Signal**

To verify the proper operation of the model, Figure 31 shows the time domain data of both the input vibration and the output waveforms. The output waveform closely approximates the analog input waveform. The quantization and jitter both cause slight distortions in the output data.

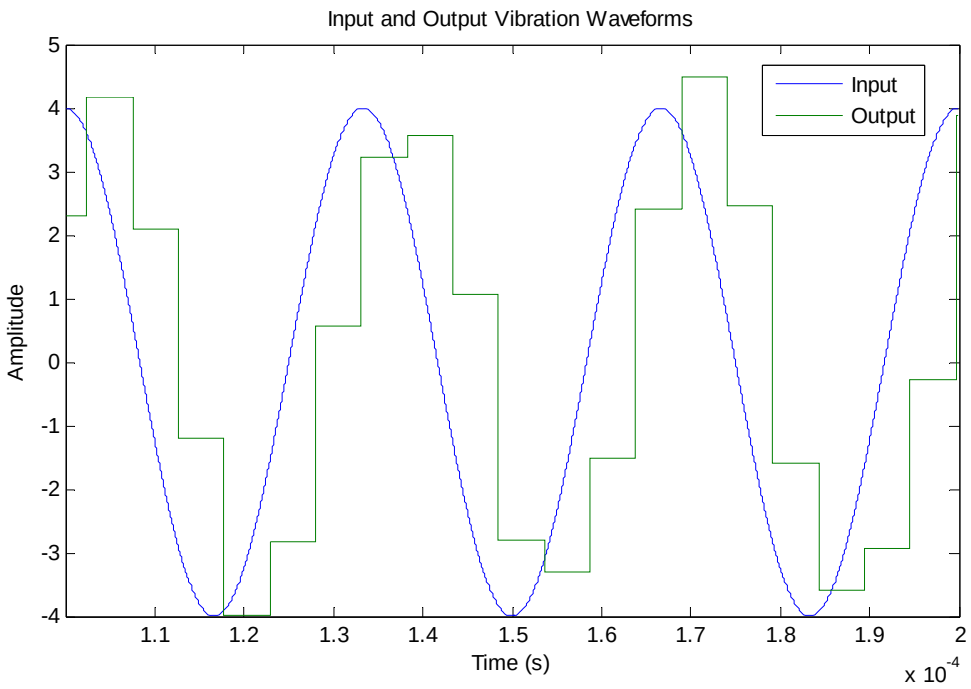



**Figure 31 - Simulated Comparison of Input vs Output Vibration Data**

## *Noise Modeling and Simulation*

As with most systems, our goal is to obtain the highest possible signal-to-noise ratio (SNR). The best SNR for a given system is when shot noise dominates all the other noise sources [8]. In a heterodyne system, a sufficiently powerful local oscillator (LO) is used to approach the quantum limit. The effects of the electronic noise from the high-gain preamplifier are overcome by using the powerful LO. However, the relative intensity noise (RIN) from the laser should still be accounted for since it is relatively strong at lower frequencies.

To compute the quantum limit, we need to determine the spectral responsivity of the photo-detector. Given a quantum efficiency of 80%, and a wavelength of 1550 nm, the spectral responsivity can be calculated as follows:

$$\rho = \frac{\eta e \lambda}{hc} = \frac{(.8)(1.6E-19)(1550E-9)}{(6.626E-34)(3E8)} = .998 \tag{27}$$

Using the responsivity calculated in Equation (27), and assuming the incoming optical power is 100 nW, the photo-detector current is:

$$I = \rho P = (.998)(100E-9) = 99.8nA \tag{28}$$

In order to calculate the SNR, the square-filter bandwidth is computed by integrating the filter response and dividing by the maximum gain of the filter. This calculation gives a square-filter bandwidth of 8.24 MHz. Equation (29) shows that the SNR is then determined using the current from Equation (28) and the square-filter bandwidth, B.

$$SNR = \sqrt{\frac{I}{2eB}} = \sqrt{\frac{99.8n}{2(1.6E-19)(8.24M)}} = 194.54 \rightarrow 45.78dB \tag{29}$$

The spectral composition of shot noise is a white spectrum given by Equation (30).

$$S_{NS}(f) = \sqrt{2eI} = \sqrt{2(1.6e-19)(99.8n)} = 1.787e-13 \, A/\sqrt{Hz} \tag{30}$$

Equation (30) shows that for all spectral components, the noise is a constant that is dependent upon the average photo-detector current.

However, the SNR calculated in Equation (28) is for an ideal system. In the actual system, the RIN from the laser must also be taken into account. The RIN was measured using a spectrum analyzer with the following setup characteristics:

| | |
|---|---|
| $Z = 4 \cdot 10^4$ V/A | Trans-impedance gain of the detector |
| $R_{in} = 50\ \Omega$ | Input impedance of the spectrum analyzer |
| $P_{RIN\text{-}Meas} = 28.3\ \mu W$ | Optical laser power incident on the detector |

We generated a line-fit shown in Equations (31) and (32) using the measured data.

For frequencies < 1 MHz:

$$dP = 2.05 \cdot 10^{-15} \cdot 77.45^{f} \tag{31}$$

For frequencies ≥ 1 MHz:

$$dP = 2.89 \cdot 10^{-16} + \frac{0.017775}{5.625e9 + (f \cdot 10^6 - f_{RIN-Peak} \cdot 10^6)^2} \tag{32}$$

where $f_{RIN\text{-}Peak} = 1.32$ MHz, measured RIN maximum location.

The measurement units then need to be converted from W/Hz to $A/\sqrt{Hz}$ which are calculated based on the circuit setup. The current from the photo-detector is converted to a voltage with the trans-impedance amplifier having a gain of Z. This voltage is measured by the spectrum analyzer as a power based on the input impedance $R_{in}$, which equates to $P = \frac{V^2}{R_{in}}$. To convert this power back to a current, the operation is reversed, $I = \frac{\sqrt{P \cdot R_{in}}}{Z}$. Since this RIN current is based on input optical power of $P_{RIN\text{-}Meas}$, it must be translated to the equivalent power used in the shot-noise calculation. Based on the responsivity of the photo-detector, the input RIN current is 28.24 μA. Since RIN is proportional to the square-root of the average current, the square-root of the input RIN current is divided out and the result is multiplied by the square-root of the average current used in the shot-noise calculation.

The other possible significant noise source that must be analyzed is the noise from the pre-amplifier stage because it has such a large gain (20,000). The spectral noise from the pre-amplifier input is around 3 $pA/\sqrt{Hz}$. The pre-amplifier noise is modeled as white noise. Figure 32 shows a plot of the full noise spectrum. As expected, the peak of the noise spectrum is located at 40 MHz and the effects of the RIN from the laser are negated. Figure 32 also shows that the pre-amplifier noise is about 20 dB below the shot-noise limit.

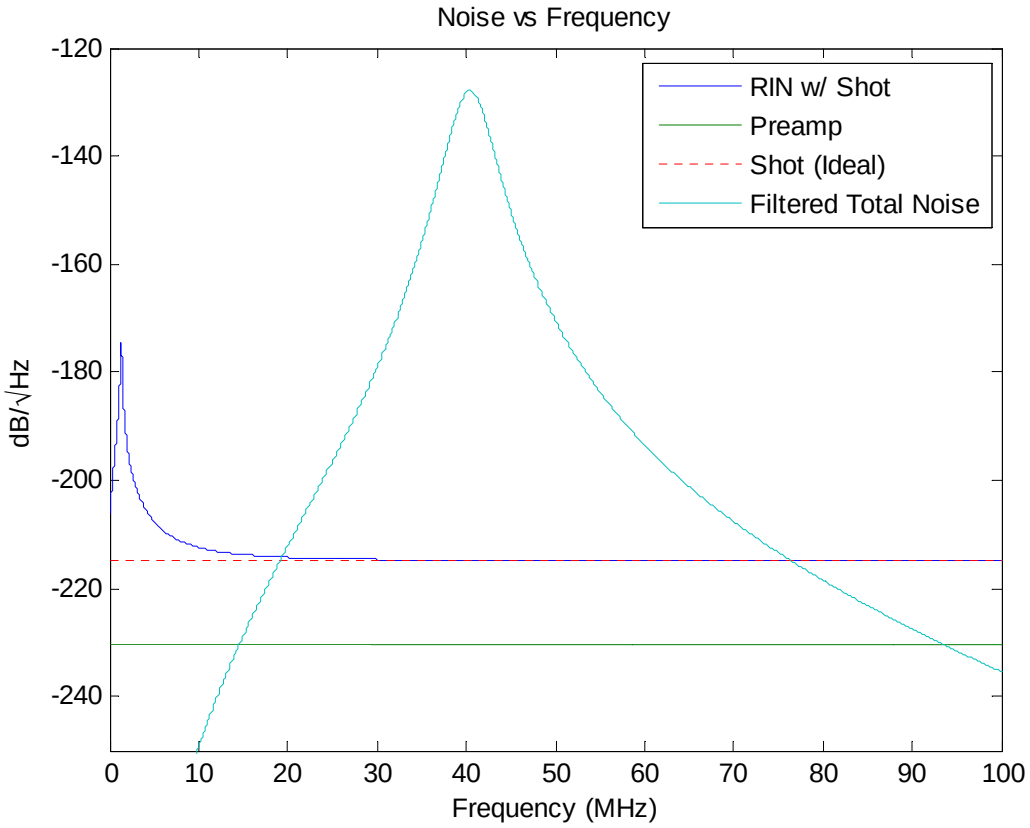



**Figure 32 - Full-Spectrum Simulated Noise Plot**

The sampled noise spectrum is determined after the filtered noise spectrum is calculated over the range of frequencies which have a significant noise component. Due to aliasing, the noise spectrum is divided into sections of Fs/2 or 6.25 MHz. The noise from each section contributes to the total noise seen in the sampled spectrum which can only represent frequencies from DC to 6.25 MHz. Also, because of the nature of aliasing, every other section is 'folded' or has a reversed spectrum contribution. Since we are aliasing, and the noise sources are considered to be independent, the RMS value for the noise is calculated in order to determine the total noise spectrum. Figure 33 shows the aliased noise spectrum.

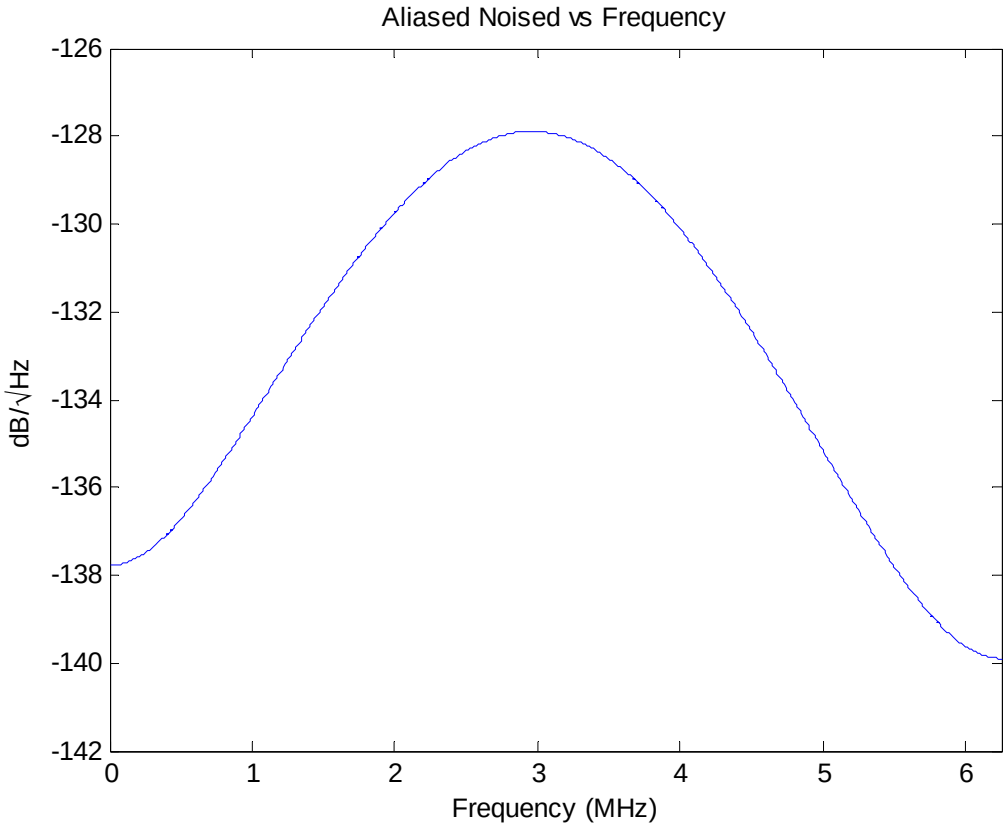



**Figure 33 - Aliased Spectrum Simulated Noise Plot**

# VIII. Implementation

One goal of this thesis is to compare the traditional over-sampling system to an under-sampling system. Thus the optics and electronics for both systems were implemented in hardware, to make a fair comparison.

## *Over-sampling Optical Test Setup*

The optics for each setup is slightly different. Figure 34 shows the over-sampling test setup where the reference beam from the laser is sent through two acousto-optic modulators (AOMs). Since the desired carrier frequency in this system is 4 MHz, one AOM must first shift the signal up by 55 MHz, and the other down by 51 MHz for a net shift of 4 MHz. Individual AOMs only work within a specific frequency range, and that range typically starts at around 40 MHz. This setup requires two independent RF drivers to power the AOMs, and the drivers must be phase-locked to each other. To generate the 4 MHz signal as input to the phase-locked loop (PLL), a mixer was required to connect to the low power outputs of the RF drivers.

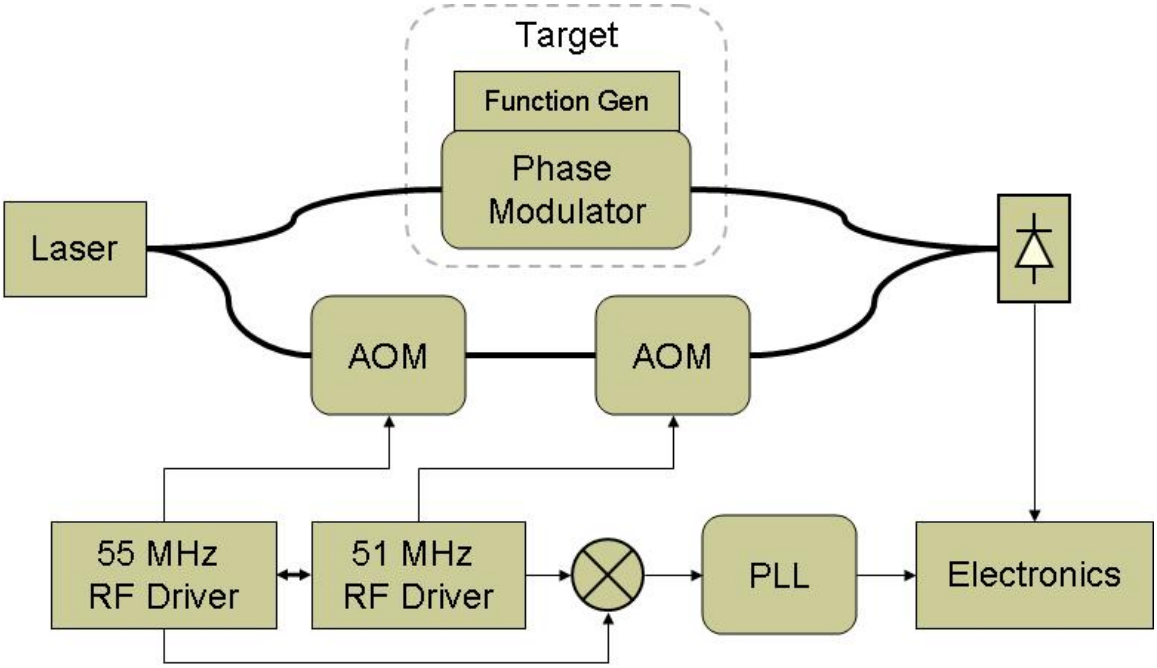



**Figure 34 - Over-sampling Optical Test Setup**

## *Under-sampling Optical Test Setup*

The optics portion of the under-sampling test setup is more straight forward than the over-sampling setup. Because the carrier is specified to be 40 MHz, only a single AOM is required. This simplification removes the need for both the second RF driver and the mixer. Figure 35 shows the diagram for the under-sampling optical test setup.

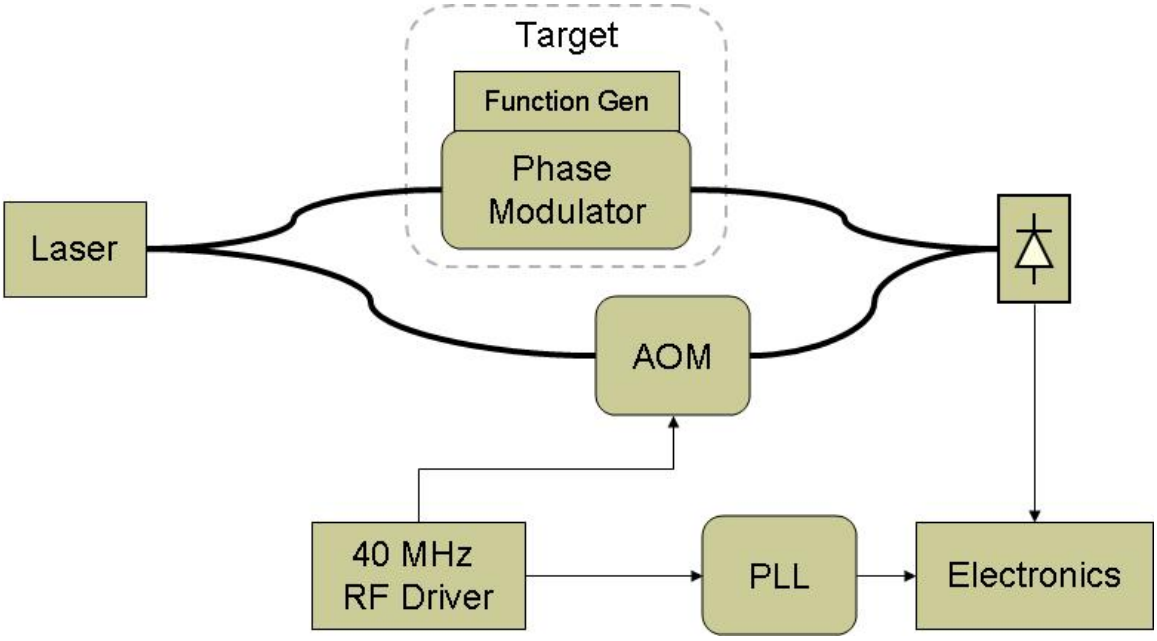



**Figure 35 - Under-sampling Optical Test Setup**

There are some additional components that the block diagrams of Figure 34 and Figure 35 do not show. These components consist of optical splitters to reduce the optical power levels, as well as fiber optics adapters. Figure 36 is a photograph of the actual optical test setup which includes the AOMs for both optical setups. The setup was created to allow for an easy transition between the under-sampling and over-sampling optical path.

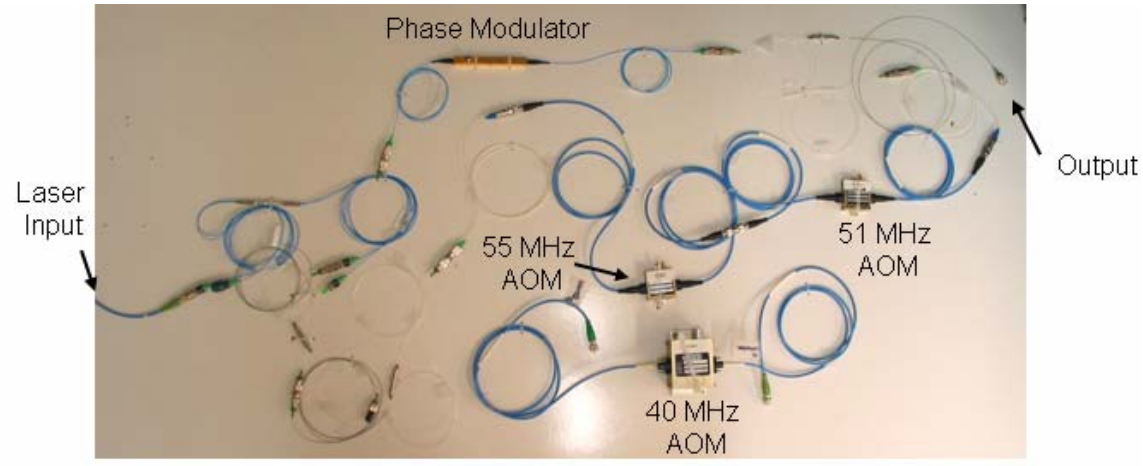



**Figure 36 - Photograph of Optical Test Setup**

## *Electrical Hardware Overview*

The laser vibrometry system features 36 independent detection channels. Because of the number of channels required, identical hardware components are placed in parallel which allows for many independent processing channels to handle all of the incoming data. Figure 37 shows an overview of the electrical hardware.

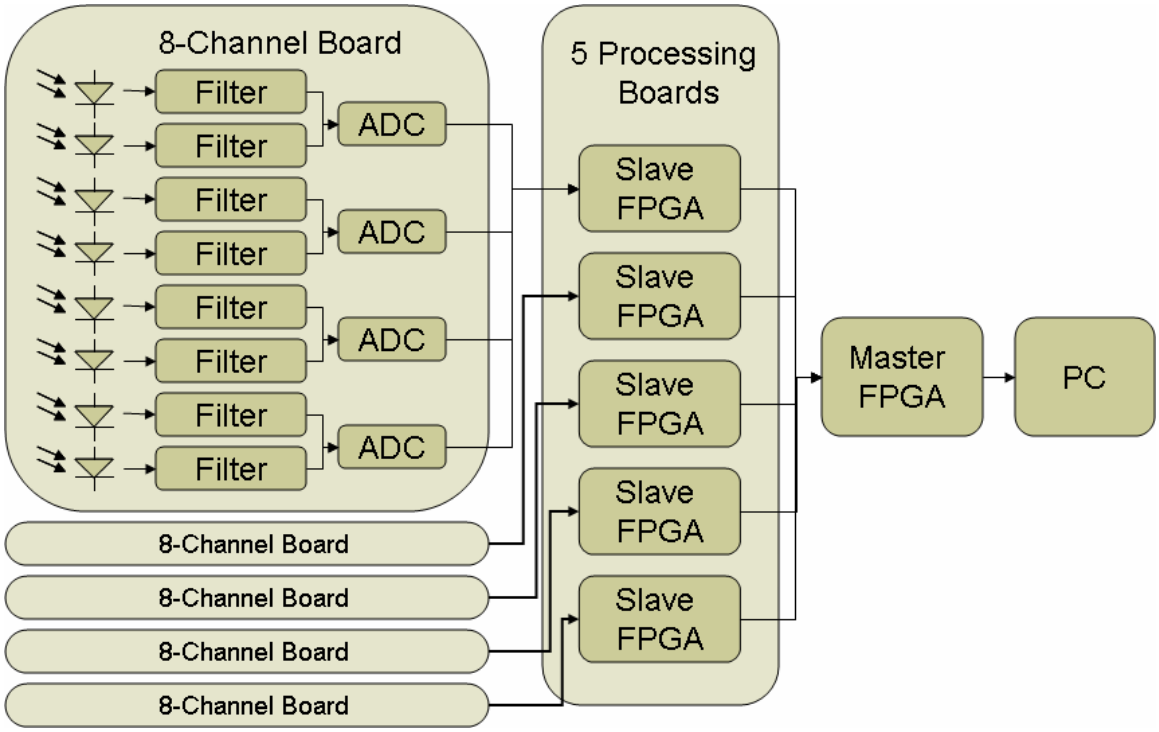



**Figure 37 - Electrical Hardware Diagram**

There are five mixed-signal, eight-channel boards to process the incoming photo-detector current. This setup supports 40 independent channels, however only 36 channels are used. The signals are filtered and amplified on eight independent channels and then sampled by four dual-channel ADCs on each board. Each eight-channel board feeds the digital signals to a slave FPGA processing board, where digital-down conversion, filtering, and decimation occur. There are five of these slave FPGA boards in the system, one for each mixed-signal board. Each slave FPGA board passes all of its data streams to the master FPGA board, where the data is processed to be sent to a computer. The computer then performs the demodulation of the I and Q streams for each channel and stores the recorded data for further processing, such as displaying the Fast Fourier Transform (FFT). In the test setup, only two channels were used, one for under-sampling

and one for over-sampling. Thus only one eight-channel analog board was required as well as one slave FPGA board and one master FPGA board. Figure 38 shows a photograph of the electronics setup. In the test setup, the output of the master FPGA is captured by a logic analyzer and then demodulated using MATLAB.

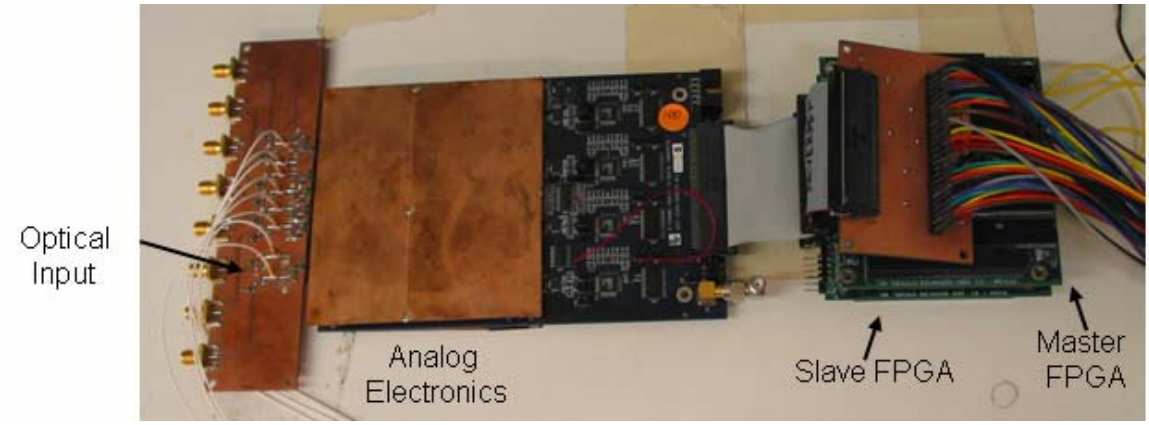



**Figure 38 - Photograph of Electronics Test Setup**

## Analog Electronics

The analog portion of the electronics circuitry consists of an Analog Devices AD8015 trans-impedance amplifier with a fixed gain of 20,000, followed by three interleaved filter and gain stages. Figure 39 shows the block diagram of this configuration.

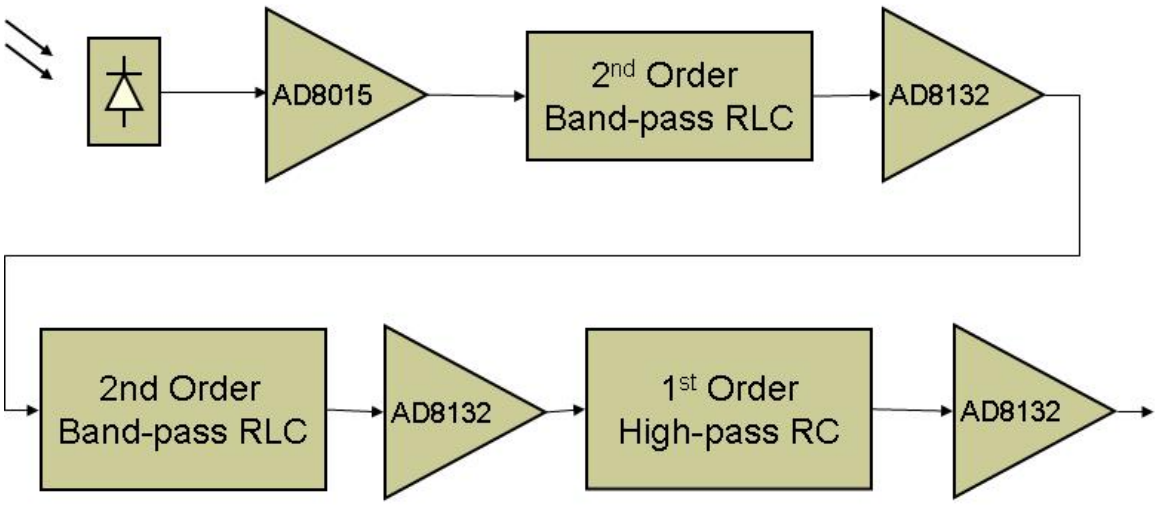



**Figure 39 - Analog Electronics Block Diagram**

The band-pass filters are second-order and each composed of a resistor, capacitor, and shielded inductor in series to simplify the design process. The high-pass filter is first-order with a series resistor and capacitor. The three gain stages use Analog Devices AD8312 differential op-amps, with a gain-bandwidth product of 320 MHz to accommodate the 40 MHz carrier signal.

## Digital Electronics

The output from the final gain stage is sent into an analog-to-digital converter (ADC) and then to an FPGA where it undergoes a series of processing steps (Figure 40). The slave FPGAs handle a significant portion of the signal processing work. Each of the five slave FPGAs process eight channels of data simultaneously. Figure 40 shows an overview of the primary processing architecture of the slave FPGA.

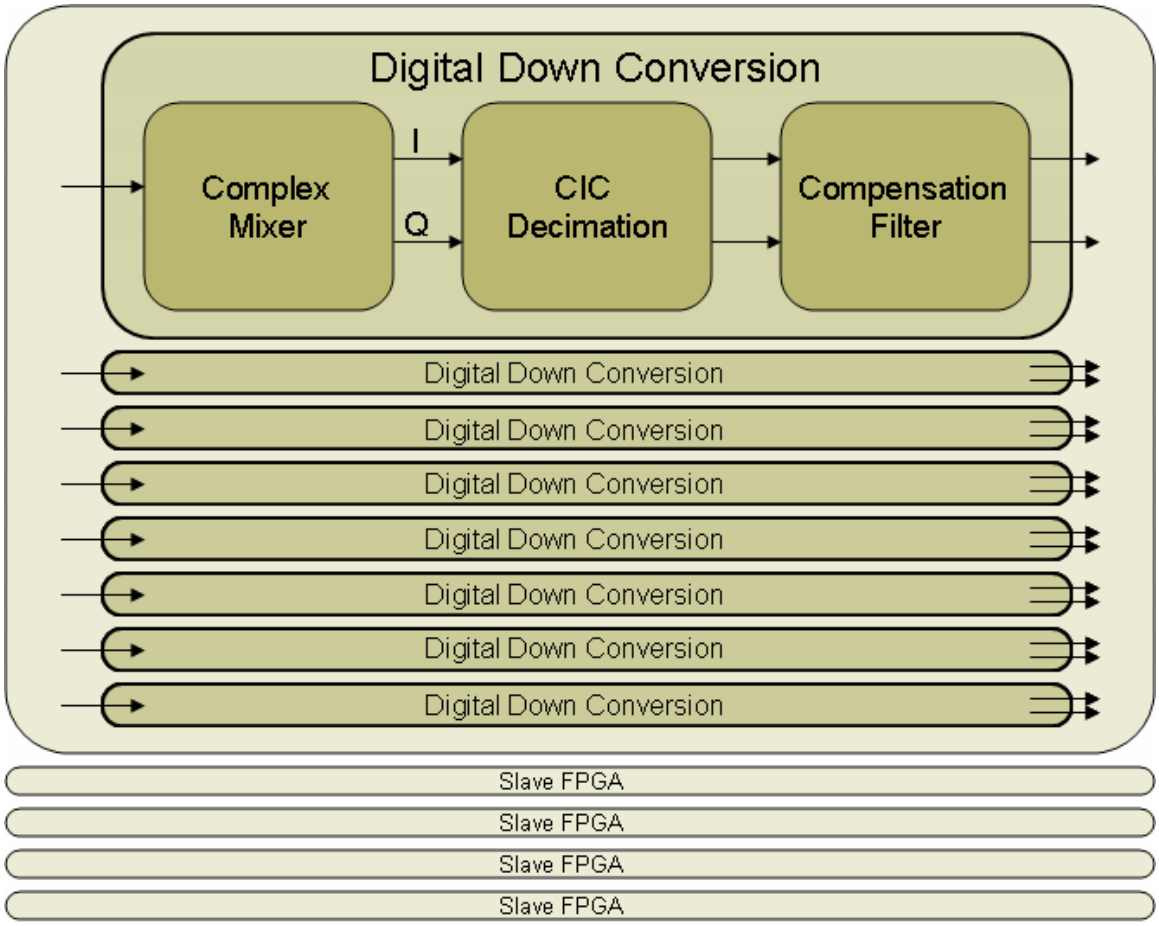



**Figure 40 - Slave FPGA Architecture**

## ADC Bus De-multiplexer

The ADCs are dual-channel and allow two analog signals to be sampled per device. However, the sampled data from both channels are interleaved onto the same ADC data bus sent to the FPGA. When the 12.5 MHz system clock signal is high, the data from channel A is sent, and when the clock signal is low, the data from channel B is sent. Figure 41 shows the timing diagram for the ADCs.

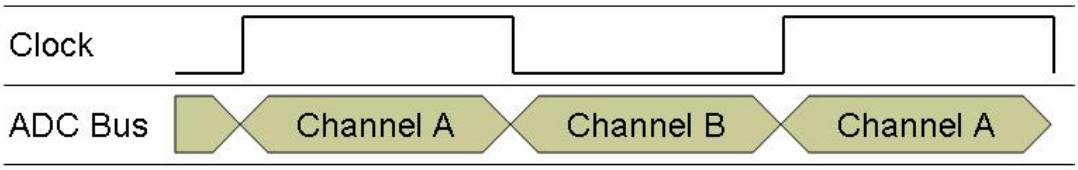



**Figure 41 - ADC Data Bus Timing Diagram**

Both channels of data are separated so they can then be digitally down-converted. Since the FPGA architecture does not support negative edge-triggered flip-flops, we designed a method for capturing the data when the clock is low. First the normal system clock is inverted and the output of the inverter is tied to a global clock line, which is internal to the FPGA. The global clock line within the FPGA is a signal trace that is physically distributed throughout all the synchronous resources on the FPGA [29]. The data from the ADC is fed to two different registers, one register uses the standard system clock for capturing data, and the other uses the inverted clock. This setup allows data to be captured when the clock goes low. The outputs of the first stage of registers feed into another set of registers which are both synchronized to the original system clock. Figure 42 shows a diagram of the de-multiplexing system.

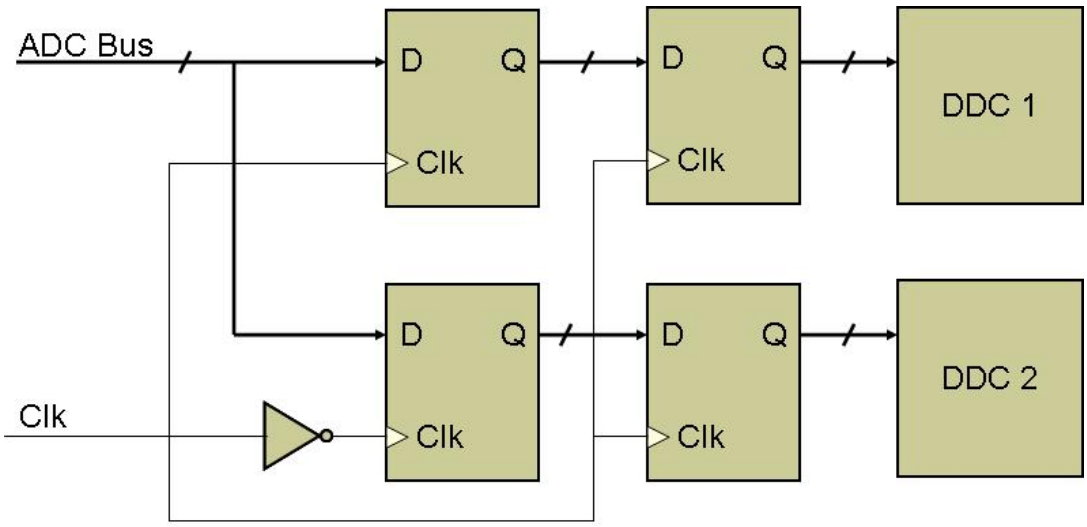



**Figure 42 - ADC Data Bus De-multiplexer Diagram**

## Complex Mixer

Once the ADC channel data is separated, the data stream is sent through a complex mixer to shift the signal of interest down to base-band. The complex mixing operation generates both an in-phase and quadrature-phase component which is useful during the demodulation process for removing amplitude variations caused by atmospheric turbulence. A numerically controlled oscillator (NCO) generates a reference sinusoid. The NCO is implemented in hardware using a block RAM which contains the predetermined amplitude values of the chosen sinusoid, based on the current phase position. The block RAM [29] is a resource available in the FPGA to store relatively large amounts of data, to avoid wasting the distributed D flip-flop resources. The phase position is incremented on the rising edge of every clock cycle. Figure 43 shows the block diagram of the complex mixer.

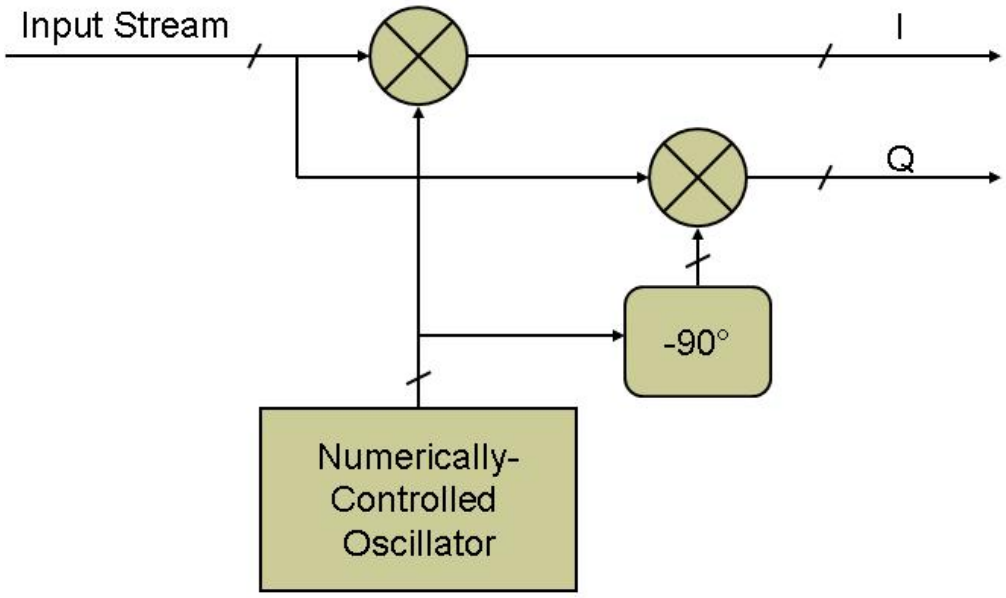



**Figure 43 - Complex Mixer**

## Cascaded Integrator-Comb Filter

The Cascaded Integrator-Comb (CIC) filter reduces the sampling rate of the base-band signal. Table 1 shows the parameters used to setup the CIC filter.

| Parameter | Value |
|---|---|
| Decimation Rate | 64 |
| Number of Stages | 5 |
| Differential Delay | 1 |

**Table 1 - CIC Filter Parameters**

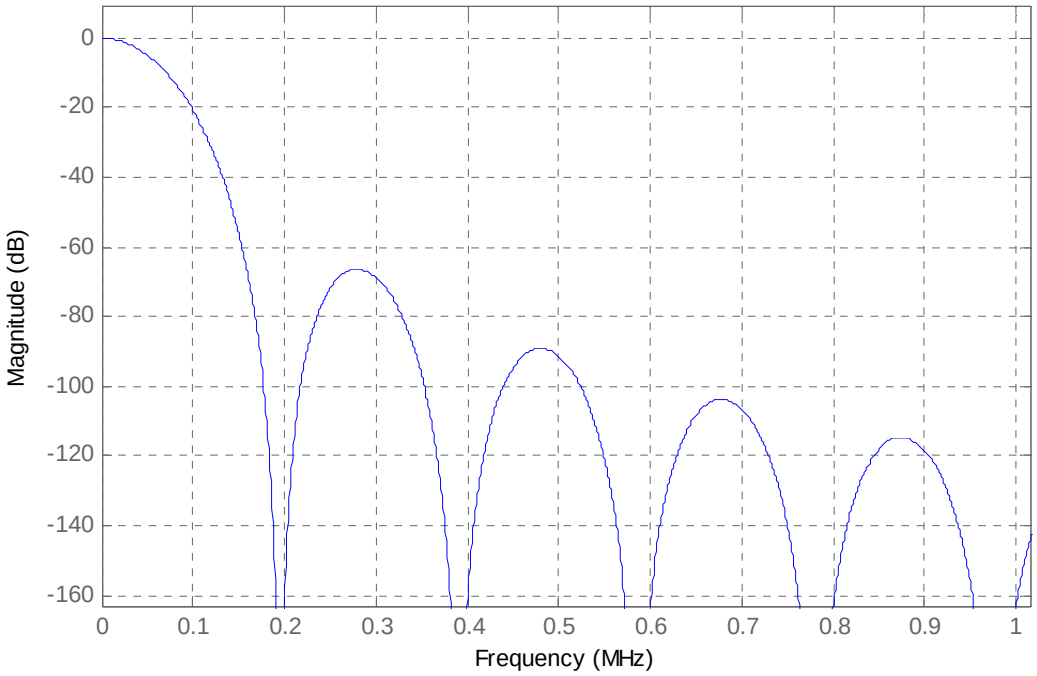



**Figure 44 - Uncompensated CIC Filter Response**

Figure 44 shows the frequency response of the CIC filter. The resulting plot closely resembles a sinc-squared function. The problem with the CIC filter is that it has a gradual transition region. Figure 45 shows an enlarged view of the pass-band filter response. The pass-band should ideally be about 20 kHz, however at 20 kHz the frequency response is attenuated by approximately 1 dB and the slope at the cut-off is relatively flat.

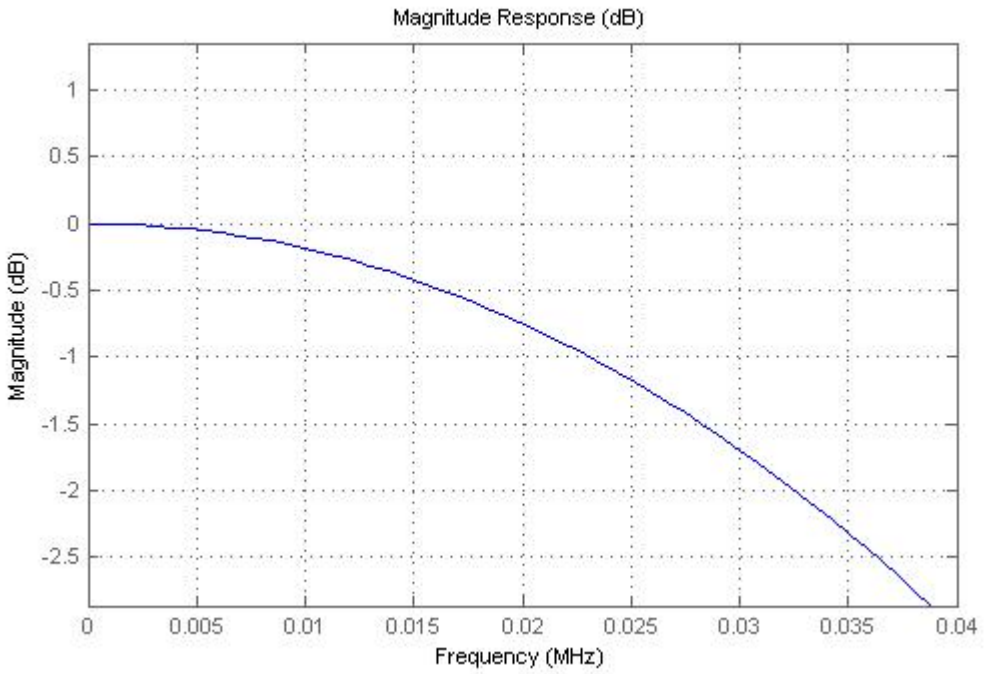



**Figure 45 - Uncompensated CIC Filter Pass-band Response**

## Compensation Filter

To counter the effects of the poor pass-band response from the CIC filter, a compensation filter is cascaded to the output of the CIC filter. In MATLAB, a compensation filter was designed to have an inverse-sinc type frequency response. In this implementation, it is setup as a thirty-tap symmetric FIR filter. The filter also further reduces the sampling rate by a factor of four, so the output sampling rate of the system is 48.8 kHz. Figure 46 shows the compensation filter and its cascaded effects. By using both the CIC and compensation filter, we achieve a flat pass-band with a sharp cut-off and a high level of attenuation in the stop band.

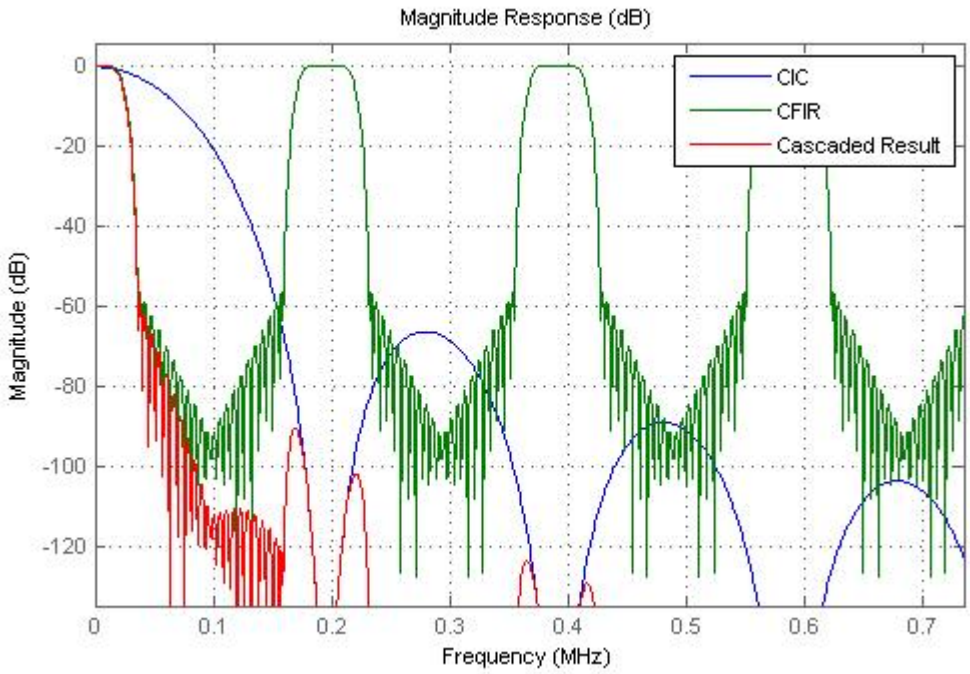



**Figure 46 - Compensated CIC Filter Response**

Figure 47 shows a zoomed-in plot of the pass-band for the compensated filter and the cascaded result. Even though the filter has twenty taps, it is implemented in hardware using a minimal amount of resources. Resource utilization is minimized because the multipliers are clocked at the output sampling rate of the DDC, which is around 48.8 kHz, so the hardware can be time-multiplexed.

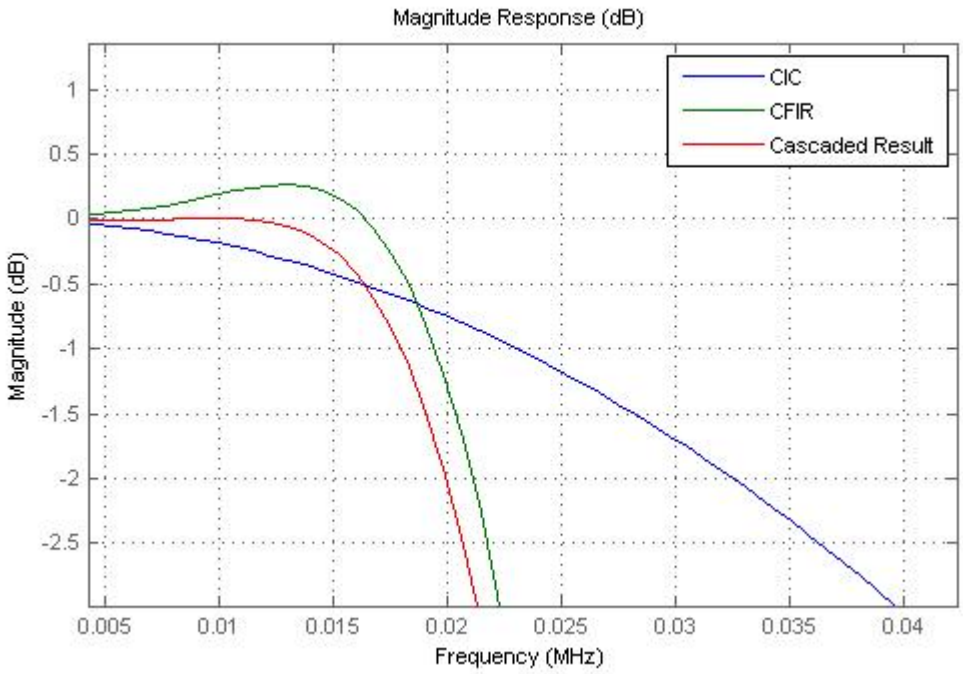



**Figure 47 - Compensated CIC Filter Pass-band Response**

## Channel Interleaver

Each of the five slave FPGA boards can process up to eight channels of complex data at a time. However, the bus to transmit the data to the master FPGA is shared by all the slave FPGA boards. Since the bus is only wide enough for one channel of complex data to be sent at a time, the data must be interleaved together. Equation (33) shows that the output rate of each DDC is equivalent to the input sample rate divided by the total filter decimation ratio (R).

$$F_{S-Out} = \frac{F_{S-In}}{R} = \frac{12.5MHz}{256} \approx 48.8kHz \tag{33}$$

The output of each DDC is buffered and the buffers feed into an 8:1 multiplexer. The multiplexer is controlled by a counter that cycles through all eight register values. The multiplexer module loads each register value onto the output bus along with the current channel identification number. The test setup uses a single eight-channel board, so the clock that feeds the counter runs eight times faster than the output rate of the DDCs, or at approximately 390 kHz. Figure 48 shows the diagram for the channel interleaver.

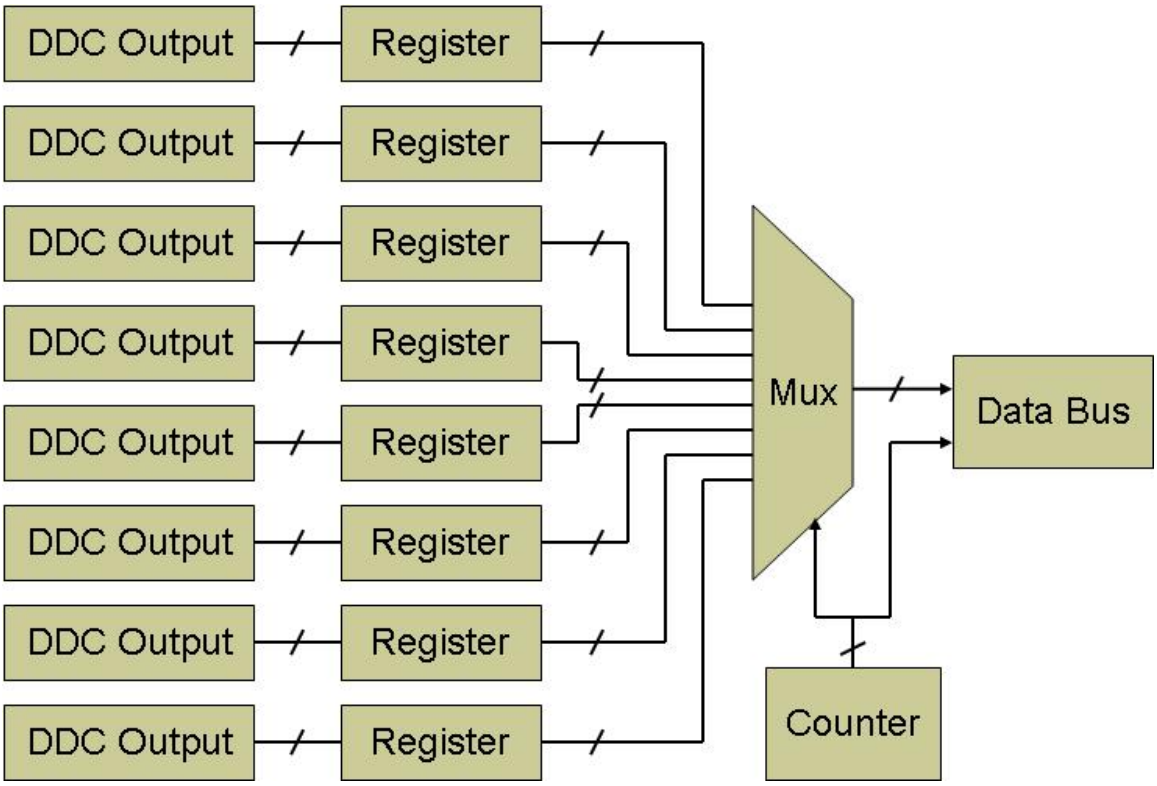



**Figure 48 - Channel Interleaver Block Diagram**

# IX. Results and Analysis

The analysis of the system is divided into multiple categories. First the optical carriers generated by the optical test setups are examined, followed by the recording of the analog inputs and digital outputs of the ADCs. Finally the demodulated outputs of the system are analyzed at different vibration frequencies.

## *Optical Carriers*

Before connecting the optics to the electronics, we examined the optical carriers that were generated for both the over-sampling and under-sampling systems. Figure 49 shows the carrier from the under-sampling system.

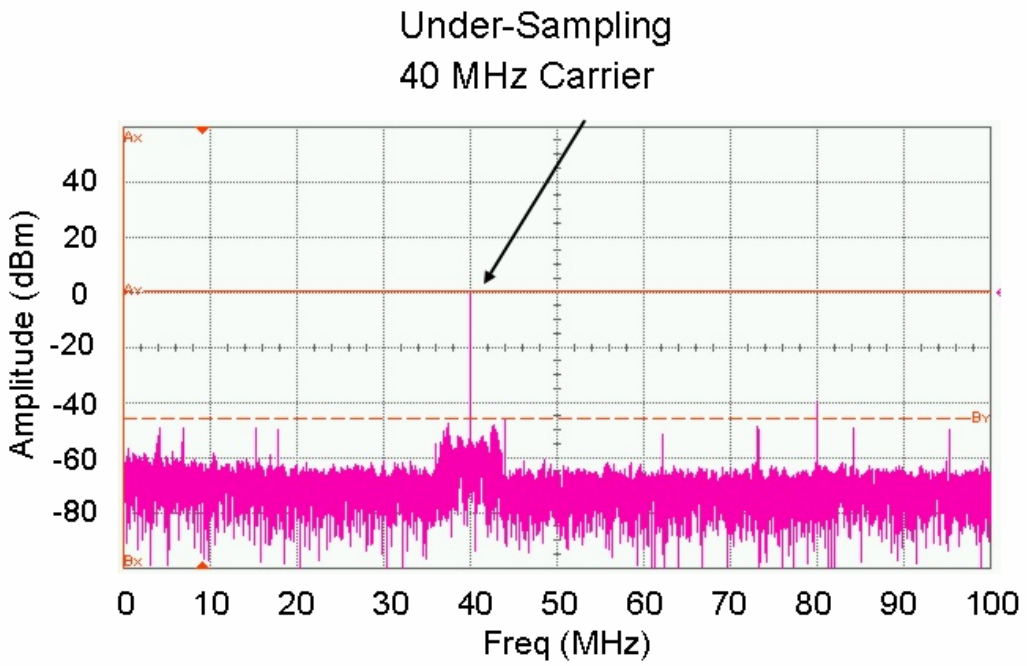



**Figure 49 - Under-Sampling Optical Carrier**

The 40 MHz carrier is a relatively strong and clean impulse at the desired frequency. Its spurious-free dynamic range (SFDR) is approximately 47 dB. Figure 50 shows the over-sampling optical carrier.

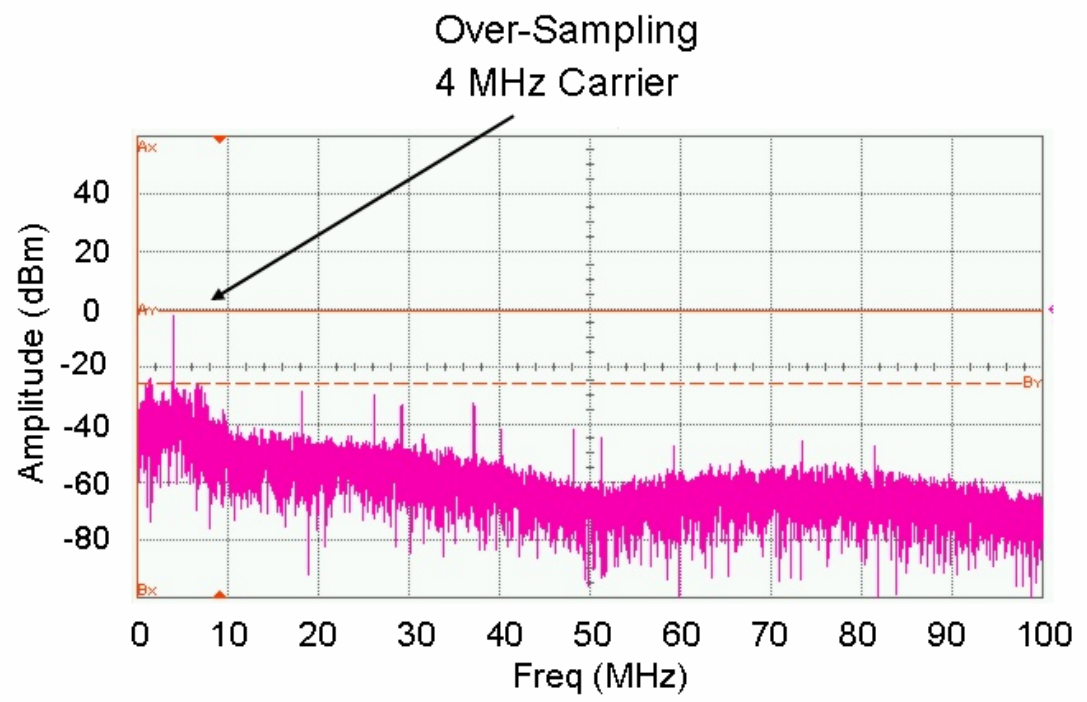



**Figure 50 - Over-Sampling Optical Carrier**

The significant characteristic we noticed when examining the over-sampling optical carrier is that the noise floor near the carrier is much higher than with the under-sampling carrier. This leads to a SFDR is approximately 26 dB, or about 21 dB less than that of the under-sampling carrier. Ideally, these two carriers should have equal performance, however the problem with the over-sampling carrier was traced back to a defective AOM. Unfortunately, a replacement AOM was not available within the given time frame. The result of this issue is that we will see in the demodulated output

spectrum that the noise floor for the over-sampling system is much higher than that of the under-sampling system.

## *ADC Inputs and Outputs*

The next section to examine in the system is the inputs and outputs of the ADC to verify that the carriers appear as expected in the electronics section. Figure 51 shows the under-sampling analog input to the ADC.

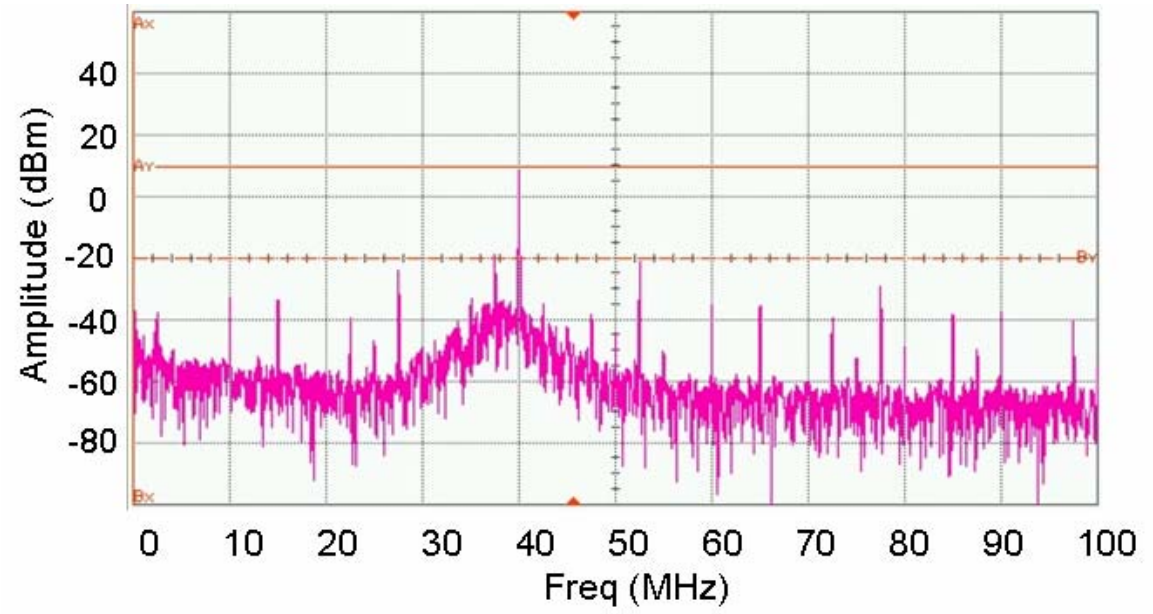



**Figure 51 - Under-Sampling ADC Input**

Several observations can be made by examining the analog input to the ADC in this setup. First, the 40 MHz carrier is strong as expected. However, we see many other relatively strong impulses at other frequencies. These impulses are due to the 12.5 MHz system clock and its harmonics, as well as inter-modulation between the clock and the

carrier frequency. The original electronics board was designed for the over-sampling system, and the clock signal harmonics were not an issue in that case because the clock frequency was much higher than the carrier frequency. Given the way the board was originally laid out, the clock signals are able to couple into the analog inputs of the ADC. Fortunately, we are only interested in a narrow bandwidth around the carrier frequency, and the other noise impulses do not significantly affect our results since we are filtering to a narrow bandwidth inside the FPGA.

One other thing we can observe is that the peak of the analog filter is slightly off from its designed center of 40 MHz. This characteristic is due to the fact that filtering at these frequencies requires low-valued capacitors and inductors. The capacitors in each filter are approximately 8 pF, and any small deviation from that value will shift the center frequency of the filter. The traces themselves and the op-amp inputs both add additional capacitance to the filter as well and shift the center frequency. Variable trim capacitors were added to the filters to help tune the center frequency as close as possible to the desired value. To verify the ADC operation, Figure 52 shows the digital output from the ADC in the under-sampling setup.

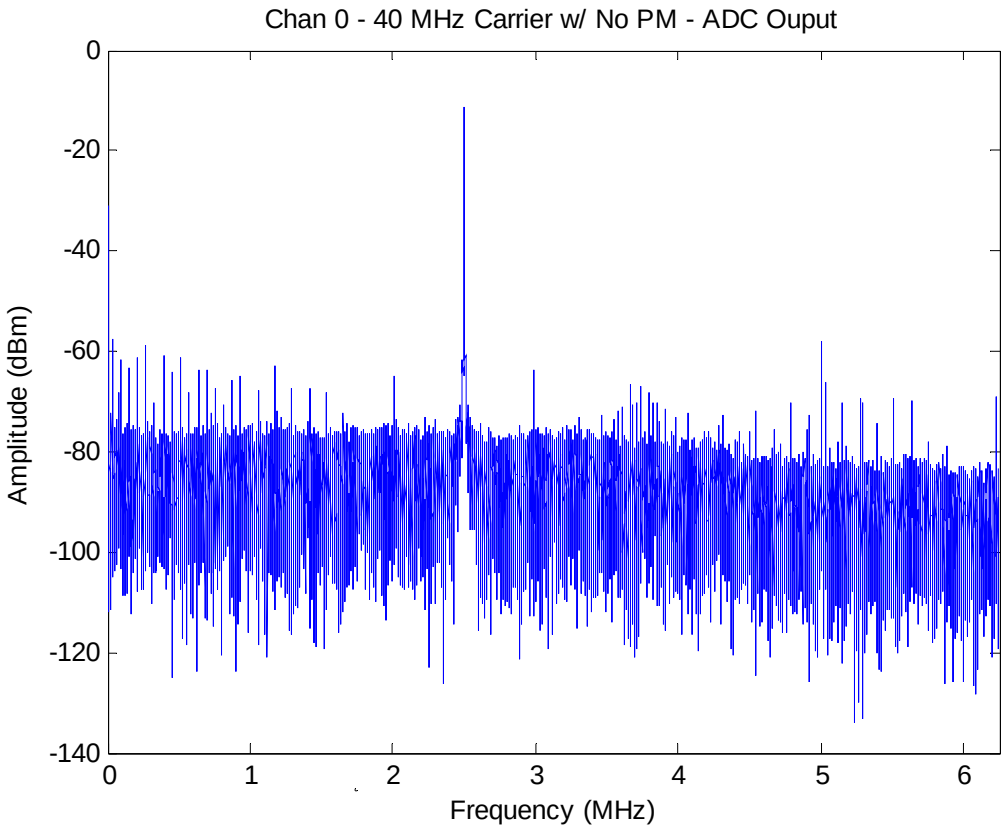



**Figure 52 - Under-Sampling ADC Output**

As expected, the 40 MHz carrier reappears at 2.5 MHz after under-sampling with a 12.5 MHz sampling frequency. Even though the analog input to the ADC was noisy, the ADC output is cleaner even before digital filtering, due to the fact that all the harmonics from the clock are under-sampled down to DC in the digital domain.

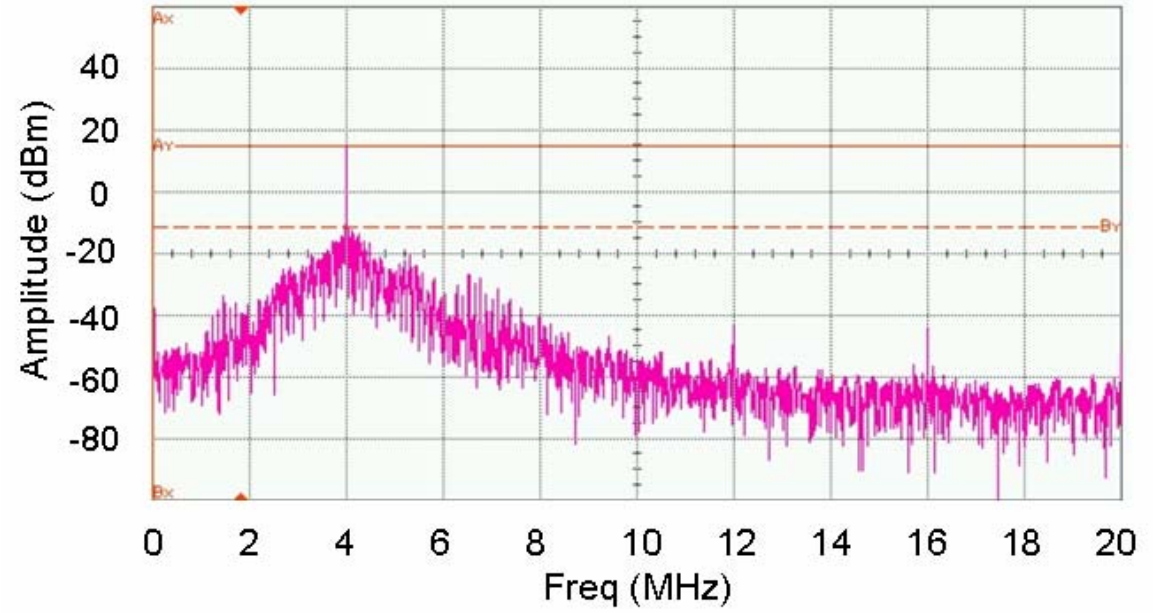



**Figure 53 - Over-Sampling ADC Input**

Figure 53 shows the analog ADC input for the over-sampling channel. The design of the over-sampling 4 MHz filter was much easier because the trace and input capacitances did not affect the center frequency of the filter as much. Figure 53 also indicates that the clock signal does not have any affect on the input since the clock frequency is 20 MHz, and its harmonics are 40 MHz, 60 MHz, etc. Figure 54 shows that the effects of the raised noise floor in the optical carrier lead to the raised noise floor in the analog ADC input and consequently on the ADC output.

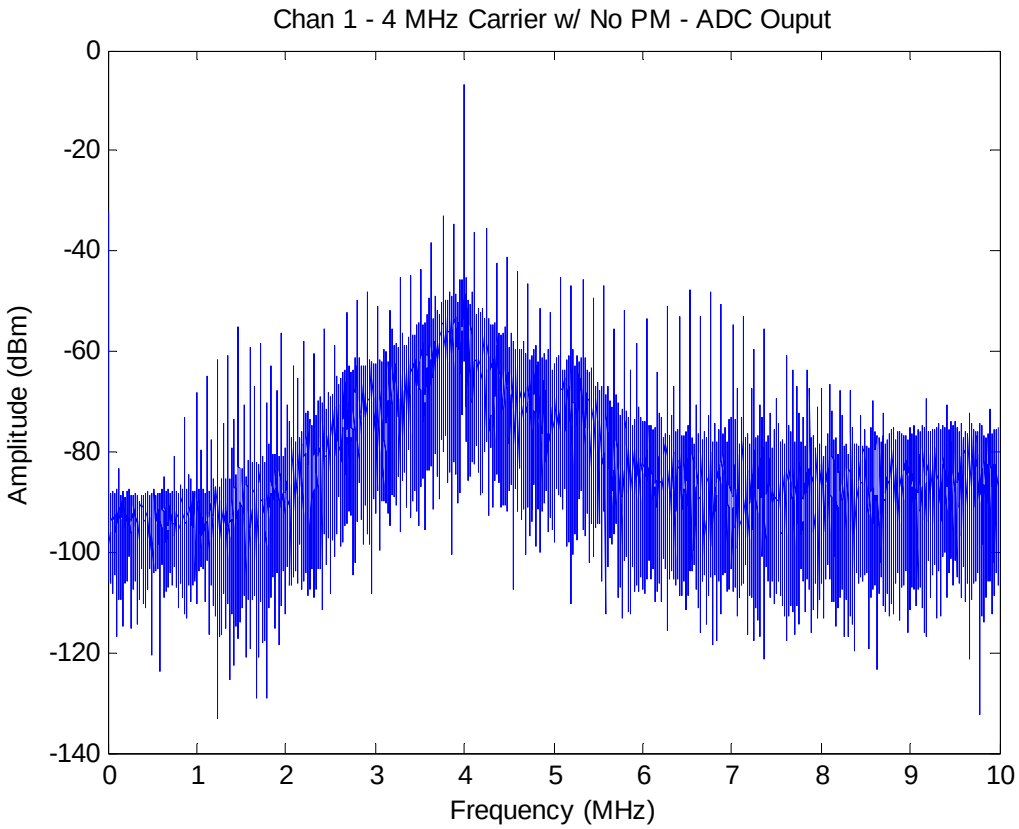



**Figure 54 - Over-Sampling ADC Output**

## *Demodulated Outputs*

The goal of the processing that occurs is to recover both the original amplitude and frequency of the vibrations on the given target. The demodulated outputs are examined in the frequency domain as well over a number of different vibration frequencies to make sure the system performs well throughout its bandwidth. First, we establish a baseline for the demodulated outputs by not phase-modulating any vibration frequency into the system. Figure 55 shows the output with no vibration.

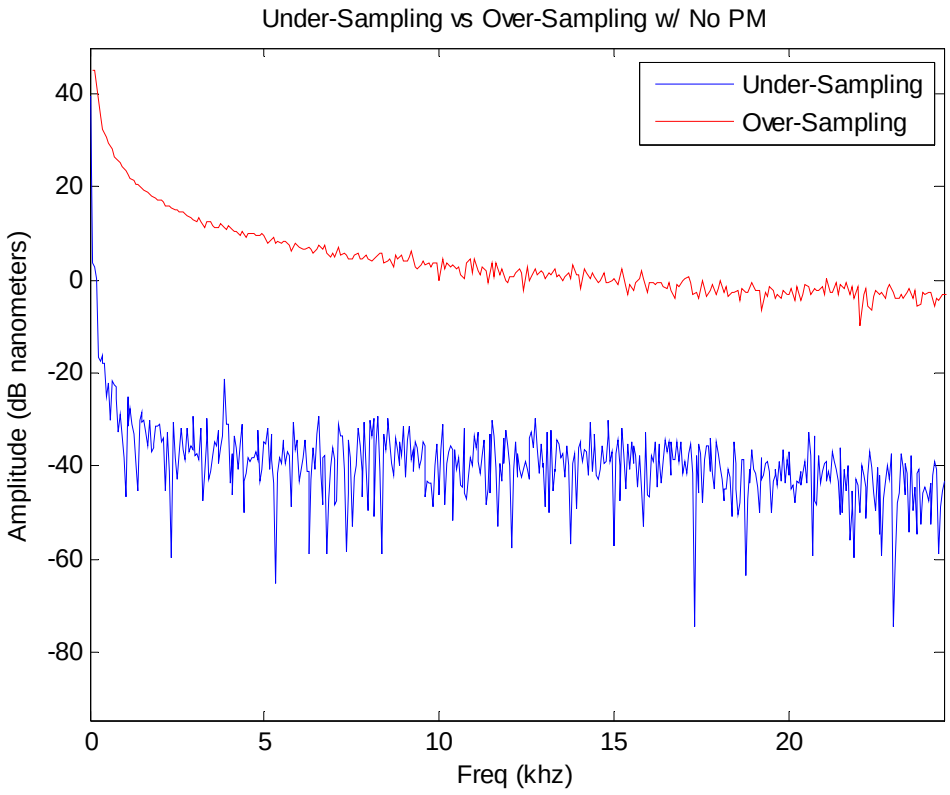



**Figure 55 - Demodulated Output with No Vibration**

Figure 55 shows that the noise floor from the under-sampling output is significantly lower than that of the over-sampling output which is caused by the over-sampling optical carrier being significantly weaker than the under-sampling carrier. If both carriers were equal, we would expect the noise floor in the over-sampling system to actually be slightly lower due to the fact the electronics were designed for the over-sampling system where jitter was not as critical an issue. In the given implementation, the under-sampling system is actually jitter-noise limited. Figure 55 also shows a slight ramp up to DC in both systems which is caused by the carrier not being a perfect sinusoid at its given frequency (and does not exist for an infinite amount of time). Spectral spreading around the down-shifted carrier is present as a result of this imperfection.

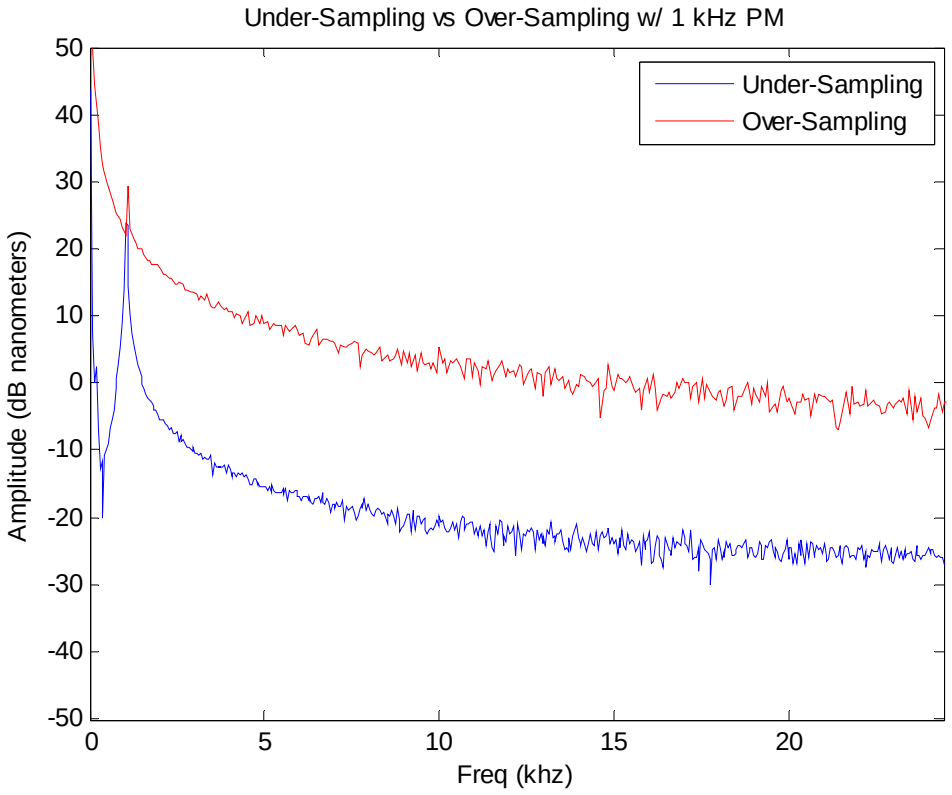



**Figure 56 - Demodulated Output with 1 kHz Vibration**

Figure 56 shows the next output is for a 1 kHz vibration frequency. Since the noise floor in the under-sampling system is much lower, the amplitude of the vibrations is much more straight-forward to identify than in the over-sampling system. Figure 57 through Figure 65 show additional system outputs for several different vibration frequencies.

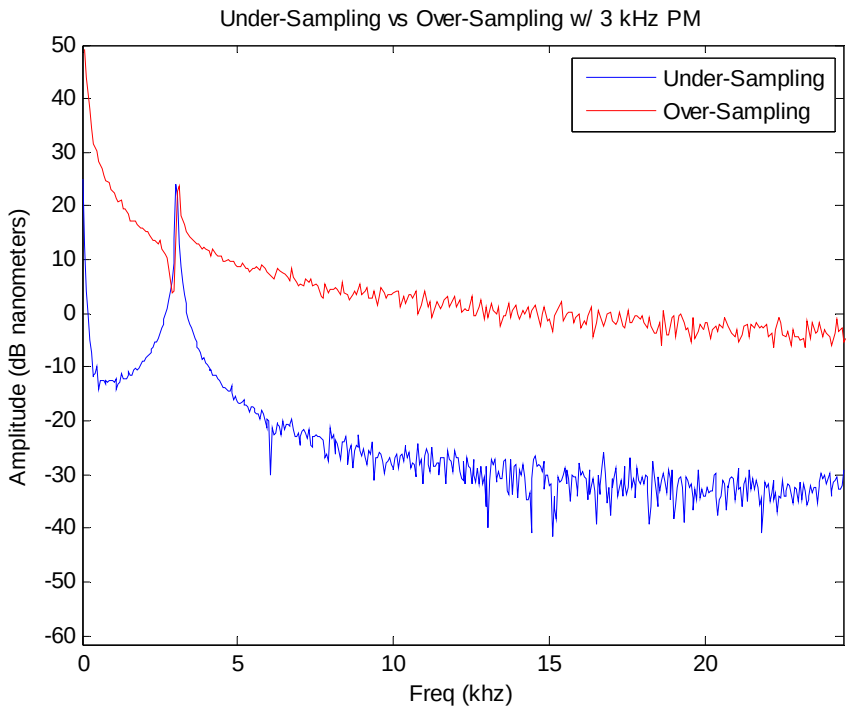



**Figure 57 - Demodulated Output with 3 kHz Vibration**

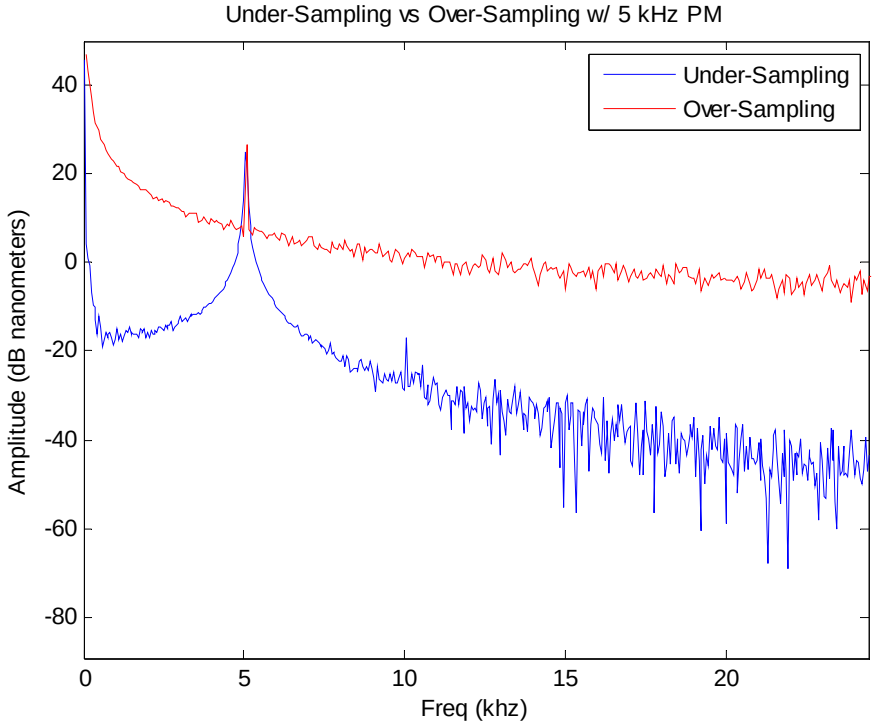



**Figure 58 - Demodulated Output with 5 kHz Vibration**

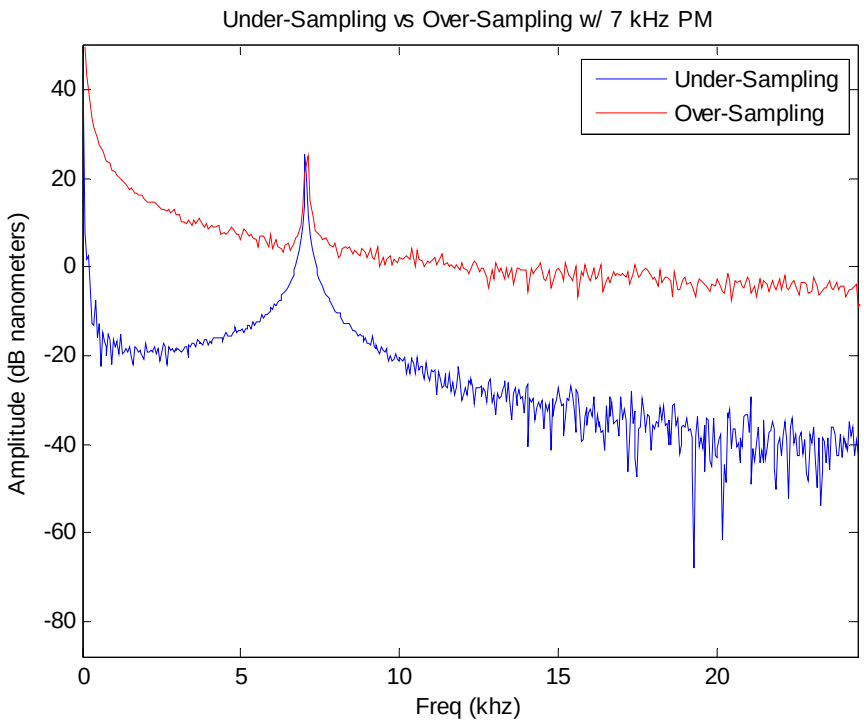



**Figure 59 - Demodulated Output with 7 kHz Vibration**

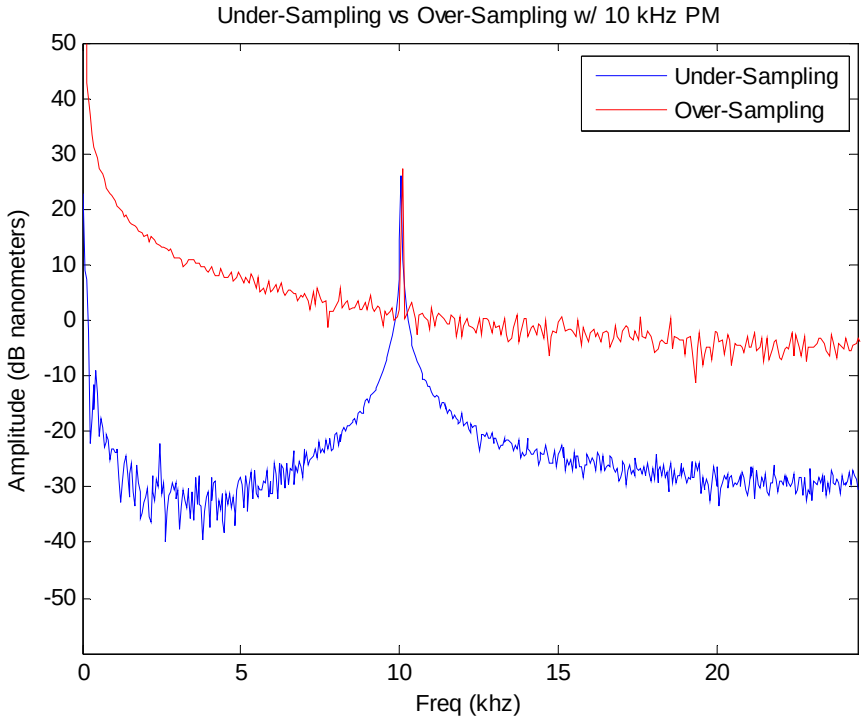



**Figure 60 - Demodulated Output with 10 kHz Vibration**

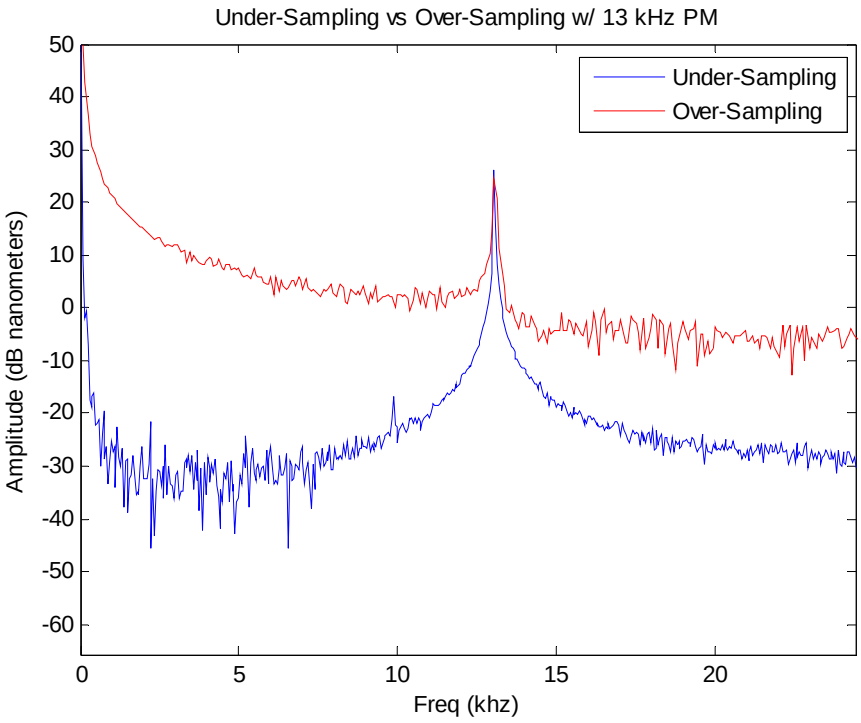



**Figure 61 - Demodulated Output with 13 kHz Vibration**

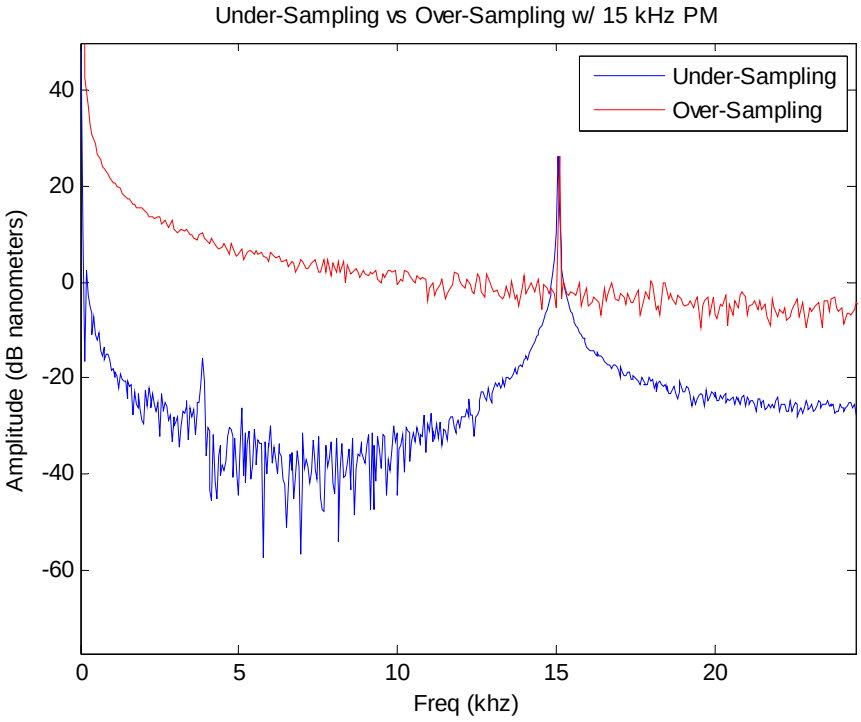



**Figure 62 - Demodulated Output with 15 kHz Vibration**

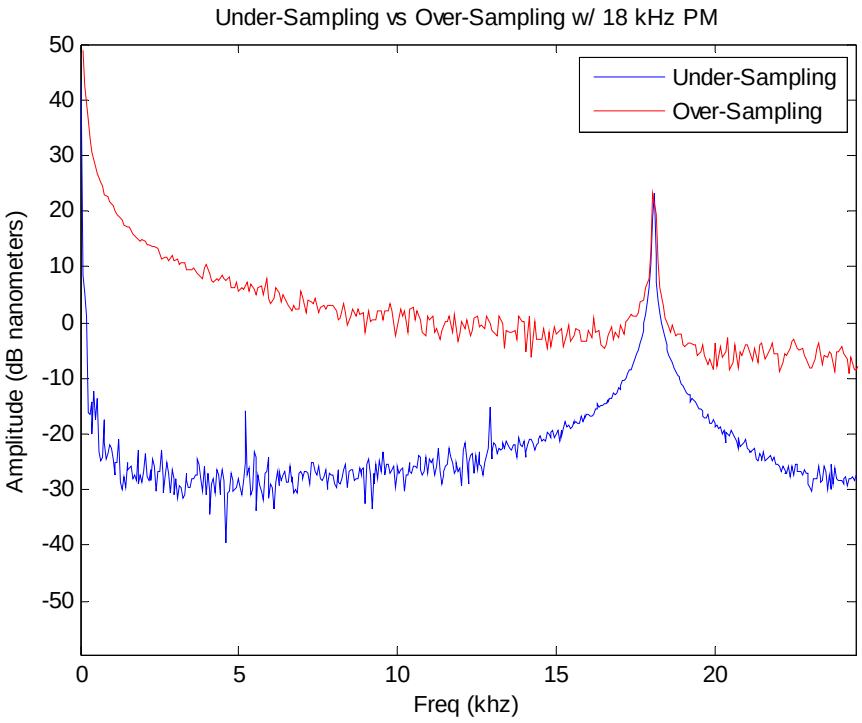



**Figure 63 - Demodulated Output with 18 kHz Vibration**

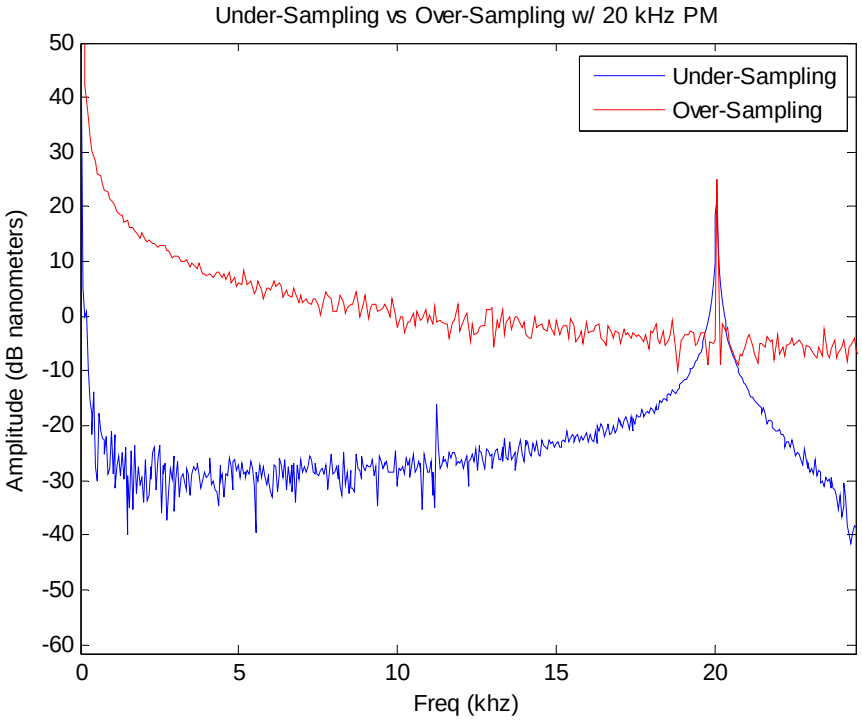



**Figure 64 - Demodulated Output with 20 kHz Vibration**

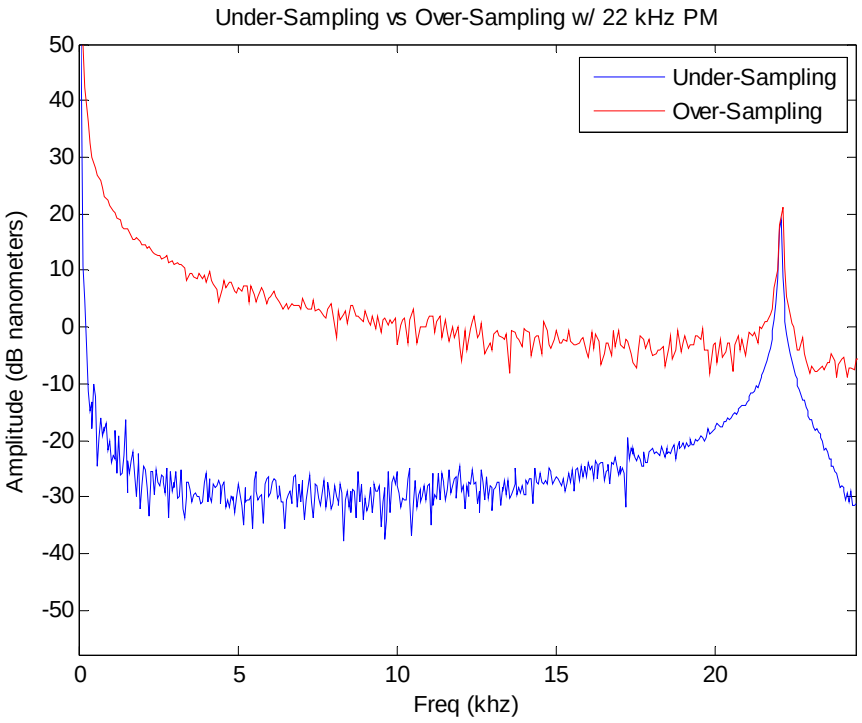



**Figure 65 - Demodulated Output with 22 kHz Vibration**

During the testing of the system at some higher vibration frequencies, we noticed there were spurs at what appeared to be random frequencies. Figure 63 and Figure 64 show these spurs. The cause of these frequency spurs was traced back to the phase-modulation. We realized that the bandwidth required for the phase-modulated vibration information depends on both the vibrations frequency and the amplitude of the vibrations. With phase-modulation, the stronger the modulation level, the further out the frequency side-lobes are extended, which contain the phase-modulation information. Since there are 36 channels running simultaneously, our goal was to minimize the amount of data produced from each channel so the bandwidth was limited to approximately 24 kHz for each channel. Thus the data stream from each channel was low-pass filtered to approximately 24 kHz to prevent aliasing, and this can end up altering the phase-

modulated information in cases where the frequency and strength of the vibrations are high enough. The result of this low-pass filtering is that after demodulation, various other spikes appear in the frequency domain. An 18° input phase-modulation strength was used in the tests performed from Figure 55 through Figure 65. To make the effects of using such a relatively narrow bandwidth more apparent, the modulation level was increased to 45° in Figure 66. However, the results from this modulation level should not be an issue in the actual system, because we expect the modulation levels to be smaller than even the 18° modulation used in the test setup. The real levels are only expected to measure a few degrees.

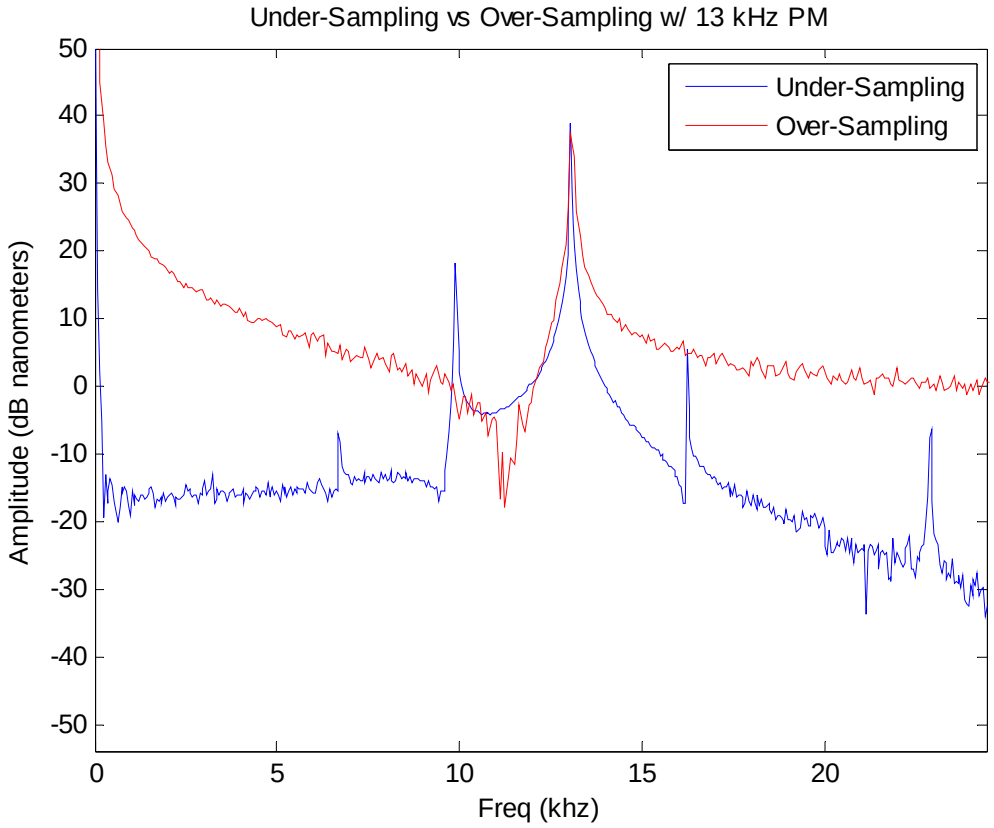



**Figure 66 - Demodulated Output with a High Modulation Level**

# X. Conclusion

The under-sampling system performed well relative to our original expectations. The SNR for the demodulated output was approximately 55 dB. We knew that because the original circuitry was designed for over-sampling, the amount of jitter added by the clock-distribution circuitry would become the limiting noise factor in the under-sampling system. As we previously discussed, the electronics were designed with the over-sampling system in mind, and the effects of jitter were not nearly as dramatic as they are for the under-sampling system. Fortunately, the clock distribution components performed better than the manufacturers' specifications, which would have limited us to an SNR of about 30 dB. We also saw that the analog input to the ADC for the under-sampling system contained a significant level of additional noise from the coupling of the clock signals into the analog inputs. There were various spurs due to both the harmonics of the system clock as well as the inter-modulation between the system clock and the 40 MHz carrier.

In the over-sampling system, we saw that the limiting factor was actually a faulty AOM in the optics setup. This defective AOM resulted in a noise floor that was approximately 20 dB higher for the over-sampling system than the under-sampling system. A replacement AOM was not available for the experiment and thus the defective AOM was used. The effects of this noisy carrier propagated through the system to the output. When the optical problem is resolved, the noise floor of the over-sampling system should only end up slightly higher than the noise floor for the shot-noise limited, under-sampling system. The theoretically higher noise floor for the over-sampling is a result of being limited by the relative intensity noise generated by the laser.

Also, the under-sampling system required only one AOM rather than two to generate the shifted reference beam. The result of this reduction was a power savings of nearly 50 watts. Given the requirements to minimize the amount of system vibration from components such as cooling fans, the resulting thermal savings from using only one AOM is very beneficial.

Overall, the novel approach of designing and implementing a multiple-channel, under-sampling system was a success. The findings from this research show that if the system is designed properly, under-sampling is a viable and advantageous alternative to traditional over-sampling in a multiple-channel laser vibrometry system. The under-sampling system is less expensive, smaller, more thermally efficient, and most importantly produces better results than its over-sampling counterpart.

# XI. Future Work

In a future revision of the electronics, the clock circuitry should be replaced with new components that have a lower level of jitter which should allow the under-sampling system to achieve the shot-noise limit. In this future layout, the ground planes and signal traces should be modified to mitigate the amount of interference on the board, which will further reduce the noise levels.

More work can be done to further optimize the digital filtering used in the slave FPGA. There are many parameters that can be modified between the CIC filter and the compensation filter including the decimation rates, the number of stages and taps, the cut-off frequencies, and various other parameters. Depending on what is deemed an acceptable level of performance, there will always be a trade-off between better performance and minimizing the amount of resources utilized within the FPGA.

Additional work can also be done in determining if a better analog filter topology exists that can be used to give a sharper band-pass cut-off around the incoming carrier signal. Having a sharper cut-off filter will generate less noise in the system output. We could conceivably add additional second-order RLC filter stages to the ADC input, but considering there are eight channels per board, it becomes a feasibility issue at that point due to space constraints and component count.

Although it will not affect the performance of the under-sampling system, the faulty AOM that was used in the over-sampling system should be replaced. This replacement will allow the vibration tests to be redone, and thus verify that the comparisons between the two systems match our expectations.

# References


[1] "Aliasing." Wikipedia, The Free Encyclopedia. 4 Jul 2006. Wikimedia Foundation, Inc. 6 Jul 2006. <http://en.wikipedia.org/w/index.php?title=Aliasing&oldid=61964799>.

[2] Ambardar, Ashok. Analog and Digital Signal Processing, 2nd Ed. New York: Thomson-Engineering, 1999.

[3] Baker, Bonnie. "Turning Nyquist Upside Down by Undersampling." Electrical Design News. 2005.

[4] Bletsis, Kelly. "Software Defined Radio." MATLAB Central File Exchange. 17 Jul 2002.

[5] Chen, Chi-Tsong. Digital Signal Processing: Spectral Computation and Filter Design. New York: Oxford University Press, Inc., 2001.

[6] "Communication Systems." Wikibooks. 28 Jun 2006. Wikimedia Foundation, Inc. 3 Jul 2006. <http://en.wikibooks.org/wiki/Communication_Systems >

[7] Donadio, Matthew. "CIC Filter Introduction." IEEE. 2000.

[8] Donati, Silvano. Photodetectors: Devices, Circuits, and Applications. Upper Saddle River: Prentice-Hall, Inc., 2000.

[9] "DSP Design Guide." Frequency Devices, Inc. Aug 2003. 30 Jun 2006. <http://www.freqdev.com/guide/dspguide.html>.

[10] Franco, Sergio. Design with Operational Amplifiers and Analog Integrated Circuits, 3rd Ed. New York: McGraw-Hill, 2002.

[11] Hosking, Rodger. "Undersampling Simplifies Wireless Control Systems." Pentek, Inc. 2005.

[12] Kester, Walt. Mixed-Signal and DSP Design Techniques. Burlington: Newnes Press, 2003.

[13] Lathi, B.P. Modern Digital and Analog Communication Systems, 3rd Ed. New York: Oxford University Press, Inc., 1998.

[14] Little, Liesl, et al. "Design Overview of the 36-Channel Vibrometry System." Lawrence Livermore National Laboratory. Report Number: UCRL-TR-217954. Dec 2005.

[15] "Measuring Jitter in Digital Systems: Application Note 1448-1." Agilent Technologies. 2003.

[16] McCormack, Paul. "Effects and Benefits of Undersampling in High-Speed ADC Applications." Design & Elektronik (Germany). 2004.

[17] "Nyquist–Shannon Sampling Theorem." Wikipedia, The Free Encyclopedia. 27 Jun 2006. Wikimedia Foundation, Inc. 28 Jun 2006 <http://en.wikipedia.org/w/index.php?title=Nyquist%E2%80%93Shannon_sampling_theorem&oldid=60922784>.

[18] Palais, Joseph. Fiber Optic Communications, 5th Ed. Upper Saddle River: Prentice-Hall, Inc., 2004.

[19] "Phase Noise Reference." Applied Radio Labs. 2003.

[20] "Putting Undersampling to Work." Pentek, Inc.

[21] Rorabaugh, Britton. DSP Primer. New York: McGraw-Hill, 1999.

[22] Rothberg, Steve and Ben Halkon. "Laser Vibrometry Meets Laser Speckle." Wolfson School of Mechanical and Manufacturing Engineer, Loughborough University. International Society for Optical Engineers.

[23] "Signal Acquisition and Conditioning." Texas Instruments. 2003.

[24] Smith, Doug. "Signals, Samples and Stuff: A DSP Tutorial (Part 1)." The National Association for Amateur Radio. 1998.

[25] "Software Radio." Transtech DSP. 2002.

[26] "Shot Noise." Encyclopedia of Laser Physics and Technology. 2 Feb 2006. 30 Jun 2006. <http://www.rp-photonics.com/shot_noise.html>.

[27] "Spectral density." Wikipedia, The Free Encyclopedia. 12 Jun 2006. Wikimedia Foundation, Inc. 29 Jun 2006 <http://en.wikipedia.org/w/index.php?title=Spectral_density&oldid=58206014>.

[28] Tsui, James. Digital Techniques for Wideband Receivers, 2nd Ed. Norwood: Artech-House, Inc., 2000.

[29] "Virtex-II Platform FPGAs: Datasheet." Xilinx, Inc. 01 Mar 2005.

[30] Weeks, Arthur. "Experimental Verification and Theory for an Eight-Element Multiple-Aperture Equal-Gain Coherent Laser Receiver for Laser Communications." Optical Society of America. 1998.

[31] Yates, Roy and David Goodman. Probability and Stoachastic Processes, 2nd Ed. Hoboken: John Wiley and Sons, Inc., 2005.

# Appendix A – MATLAB Code

## *MATLAB Noise Simulation Source Code*

```
function np_rin = noise_calc(filtergain)

% freq is measured in MHz with increments of 10 KHz
freq = (.01:.01:100);
light_avg = 100e-6; % 100 uW of incoming light
responsivity = .998; % Responsivity of the detector
I_avg = responsivity .* light_avg; % Avg current
q = 1.6e-19; % Electron charge
R_in = 50;  % Input impedance of the RF spectrum analyzer
Z = 4e4; % Transimpedance gain of the detector
BW = 4e6; % Bandwidth of the analog filter

rin_light_in = 28.3e-6; % Incoming optical power during RIN measurement

% Calculate the RIN
% f_rin_peak is the frequency of the peak level of rin noise
f_rin_peak = 1.32;

% Calculate RIN in W/Hz for an incoming light of 28.3uW
rin(1:100) = 2.05e-15 .* (77.45 .^(freq(1:100)));
rin(101:10000) = 2.89e-16 + (.017775 ./ (5.625e9 + ...
    (freq(101:10000).*1e6 - f_rin_peak.*1e6).^2));

% Adjust rin for an incoming light of 100uW
% Calculate the RIN current during measurement
rin_current_in = responsivity .* rin_light_in;
% Adjust the units from W/Hz to A/root(Hz)
rin = sqrt(rin .* R_in)./ Z;
% Adjust the measurement for 100 nA instead
rin = rin ./ sqrt(rin_current_in) .* sqrt(I_avg);
%rin = rin ./ I_avg; % Normalize
% Convert optical input power from 100 uW to 1 mW
rin = rin .* sqrt(1*10^-3) ./ sqrt(1*10^-4);
rin_db = 20 .* log10(rin);

% Calculate the Shot noise
shot(1:10000) = sqrt(2.*q.*I_avg);
%shot = shot ./ I_avg; % Normalize
% Convert optical input power from 100 uW to 1 mW
shot = shot .* sqrt(1*10^-3) ./ sqrt(1*10^-4);
shot_db = 20 .* log10(shot);

% Calculate the pre-amplifier noise

% Data sheet values
% preamp_x = 10^2 .* [0:5:100];
% preamp_y = 10^-12 .* [2.3 2.3 2.25 2.2 2.3 2.4 2.5 2.6 2.7 2.75...
%     2.8 2.95 3.05 3.15 3.2 3.3 3.5 3.65 3.75 3.9 3.95];
% preamp_xi = [1:10000];
% preamp_noise = interp1(preamp_x,preamp_y,preamp_xi);
% preamp_noise_db = 20 .* log(preamp_noise);
```

```
% Preamp Noise is 3 pA/sqrt(Hz)
preamp_noise = 3e-12;
preamp_noise(1:10000) = preamp_noise;
preamp_noise_db = 20 .* log10(preamp_noise);

 % Calculate the total noise
total_noise = sqrt(rin.^2 + preamp_noise.^2);% + shot.^2);
total_noise_db = 20 .* log10(total_noise);

% Calculate the total filtered noise
total_noise_filtered = abs(filtergain).^2 .* total_noise;
total_noise_filtered_db = 20 .* log10(total_noise_filtered);

figure(1);
plot(freq,rin_db,freq,preamp_noise_db,freq,shot_db,':',...
    freq,total_noise_filtered_db);

legend('RIN w/ Shot','Preamp','Shot (Ideal)','Filtered Total Noise',...
    'Location','northeast');
xlabel('Frequency (MHz)');
ylabel('dB/\surd{Hz}');
title('Noise vs Frequency');
axis([0 100 -250 -120]);

%Aliased Noise Calculation

% Divide up the frequency regions
for i=1:16
    if mod(i,2)==1
        section(i,:) = total_noise_filtered((i-1)*625+1:i*625);
    else
        % Flip the spectrum of the fs/2 - fs regions
        section(i,:) = total_noise_filtered(i*625:-1:(i-1)*625+1);
    end
    section(i,:) = section(i,:).^2;
end

% Sum up the noise in each region
total_aliased_noise = sqrt(sum(section));
total_aliased_noise_db = 20.*log10(total_aliased_noise);
x_axis = (.01:.01:6.25);

figure(2);plot(x_axis,total_aliased_noise_db);
xlabel('Frequency (MHz)');
ylabel('dB/\surd{Hz}');
title('Aliased Noised vs Frequency');
axis([0 6.25 -142 -126]);
```

## *MATLAB Demodulation Code*

```
function [freq0,fdata0] = demodiq(datai,dataq)

DDC_R = 256; % DDC Decimation Rate

% ADC Info
ADC.Fs = 12.5e6; % Sample rate of ADC
ADC.Vpp = 2; % Input voltage swing on ADC
ADC.Bits = 12; % Number of bits on ADC
ADC.Rin = 50; % Input resistance of ADC
ADC.VPB = ADC.Vpp / (2^ADC.Bits); % Calculate the number of volts per
bit

j = ones(8,1);
k = 1;

Separate the interleaved channel data
for i=1:length(chan)
    switch chan(i)
        case 0 % Channel 0 Data
            data0i(j) = datai(i);
            data0q(j) = dataq(i);
            j(1) = j(1)+1;
        case 1 % Channel 1 Data
            data1i(k) = datai(i);
            data1q(k) = dataq(i);
            j(2) = j(2)+1;
        case 2 % Channel 2 Data
            data2i(k) = datai(i);
            data2q(k) = dataq(i);
            j(3) = j(3)+1;
        case 3 % Channel 3 Data
            data3i(j) = datai(i);
            data4q(j) = dataq(i);
            j(4) = j(4)+1;
        case 4 % Channel 4 Data
            data4i(k) = datai(i);
            data4q(k) = dataq(i);
            j(5) = j(5)+1;
        case 5 % Channel 5 Data
            data5i(k) = datai(i);
            data5q(k) = dataq(i);
            j(6) = j(6)+1;
        case 6 % Channel 6 Data
            data6i(k) = datai(i);
            data6q(k) = dataq(i);
            j(7) = j(7)+1;
        case 7 % Channel 7 Data
            data7i(k) = datai(i);
            data7q(k) = dataq(i);
            j(8) = j(8)+1;
    end
end
```

```
l=0;
for i=1:length(chan)
    switch l
        case 0 % Channel 0 Data
            data0i(j) = datai(i);
            data0q(j) = dataq(i);
            j = j+1;
            l=1;
        case 1 % Channel 1 Data
            data1i(k) = datai(i);
            data1q(k) = dataq(i);
            k = k+1;
            l=0;
    end
end


% Downsample & Demodulate Channel 0
data0 = unwrap(atan2(data0q,data0i));

% Downsample & Demodulate Channel 1
data1 = unwrap(atan2(data1q,data1i));

% Downsample & Demodulate Channel 2
data2 = unwrap(atan2(data2q,data2i));

% Downsample & Demodulate Channel 3
data3 = unwrap(atan2(data3q,data3i));

% Downsample & Demodulate Channel 4
data4 = unwrap(atan2(data4q,data4i));

% Downsample & Demodulate Channel 5
data5 = unwrap(atan2(data5q,data5i));

% Downsample & Demodulate Channel 6
data6 = unwrap(atan2(data6q,data6i));

% Downsample & Demodulate Channel 7
data7 = unwrap(atan2(data7q,data7i));

Num_samples_0 = length(data0);
Num_samples_1 = length(data1);
Num_samples_2 = length(data2);
Num_samples_3 = length(data3);
Num_samples_4 = length(data4);
Num_samples_5 = length(data5);
Num_samples_6 = length(data6);
Num_samples_7 = length(data7);

% FFT of Channel 0
% Compute Freq range
freq0 = [1:Num_samples_0/2]./Num_samples_0.*ADC.Fs./(DDC_R .* 1e3);
fdata0 = abs(fft(data0))./length(data0); % Compute FFT and log scale
```

```
fdata0 = fdata0 .* ADC.VPB;
fdata0 = fdata0.^2 ./ ADC.Rin;
fdata0 = 10 .* log10(1000 .* fdata0);
fdata0 = fdata0(1:Num_samples_0/2);

% FFT of Channel 1
% Compute Freq range
freq1 = [1:Num_samples_1/2]./Num_samples_1.*ADC.Fs./(DDC_R .* 1e3);
fdata1 = abs(fft(data1))./length(data1); % Compute FFT and log scale
fdata1 = fdata1 .* ADC.VPB;
fdata1 = fdata1.^2 ./ ADC.Rin;
fdata1 = 10 .* log10(1000 .* fdata1);
fdata1 = fdata1(1:Num_samples_1/2);

% FFT of Channel 2
% Compute Freq range
freq2 = [1:Num_samples_2/2]./Num_samples_2.*ADC.Fs./(DDC_R .* 1e3);
fdata2 = abs(fft(data2))./length(data2); % Compute FFT and log scale
fdata2 = fdata2 .* ADC.VPB;
fdata2 = fdata2.^2 ./ ADC.Rin;
fdata2 = 10 .* log10(1000 .* fdata2);
fdata2 = fdata2(1:Num_samples_2/2);

% FFT of Channel 3
% Compute Freq range
freq3 = [1:Num_samples_3/2]./Num_samples_3.*ADC.Fs./(DDC_R .* 1e3);
fdata3 = abs(fft(data3))./length(data3); % Compute FFT and log scale
fdata3 = fdata3 .* ADC.VPB;
fdata3 = fdata3.^2 ./ ADC.Rin;
fdata3 = 10 .* log10(1000 .* fdata3);
fdata3 = fdata3(1:Num_samples_3/2);

% FFT of Channel 4
% Compute Freq range
freq4 = [1:Num_samples_4/2]./Num_samples_4.*ADC.Fs./(DDC_R .* 1e3);
fdata4 = abs(fft(data4))./length(data0); % Compute FFT and log scale
fdata4 = fdata4 .* ADC.VPB;
fdata4 = fdata4.^2 ./ ADC.Rin;
fdata4 = 10 .* log10(1000 .* fdata4);
fdata4 = fdata4(1:Num_samples_4/2);

% FFT of Channel 5
% Compute Freq range
freq5 = [1:Num_samples_5/2]./Num_samples_5.*ADC.Fs./(DDC_R .* 1e3);
fdata5 = abs(fft(data5))./length(data5); % Compute FFT and log scale
fdata5 = fdata5 .* ADC.VPB;
fdata5 = fdata5.^2 ./ ADC.Rin;
fdata5 = 10 .* log10(1000 .* fdata5);
fdata5 = fdata5(1:Num_samples_5/2);

% FFT of Channel 6
% Compute Freq range
freq6 = [1:Num_samples_6/2]./Num_samples_6.*ADC.Fs./(DDC_R .* 1e3);
fdata6 = abs(fft(data6))./length(data6); % Compute FFT and log scale
fdata6 = fdata6 .* ADC.VPB;
```

```
fdata6 = fdata6.^2 ./ ADC.Rin;
fdata6 = 10 .* log10(1000 .* fdata6);
fdata6 = fdata6(1:Num_samples_6/2);

% FFT of Channel 7
% Compute Freq range
freq7 = [1:Num_samples_7/2]./Num_samples_7.*ADC.Fs./(DDC_R .* 1e3);
fdata7 = abs(fft(data7))./length(data7); % Compute FFT and log scale
fdata7 = fdata7 .* ADC.VPB;
fdata7 = fdata7.^2 ./ ADC.Rin;
fdata7 = 10 .* log10(1000 .* fdata7);
fdata7 = fdata7(1:Num_samples_7/2);

%Plot FFT of Channel 0
figure(1);
plot(freq0,fdata0)
title('Channel 0');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 1
figure(2);
plot(freq1,fdata1)
title('Channel 1');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 2
figure(3);
plot(freq2,fdata2)
title('Channel 2');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 3
figure(4);
plot(freq3,fdata3)
title('Channel 3');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 4
figure(5);
plot(freq4,fdata4)
title('Channel 4');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 5
figure(6);
plot(freq5,fdata5)
title('Channel 5');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');
```

```
%Plot FFT of Channel 6
figure(7);
plot(freq6,fdata6)
title('Channel 6');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');

%Plot FFT of Channel 7
figure(8);
plot(freq7,fdata7)
title('Channel 7');
xlabel('Freq (khz)')
ylabel('Amplitude (dBm)');
```

## *MATLAB Filter Simulation Code*

```
% CIC Filter
CIC.Fs_in = 12.5e6;
CIC.R = 256; % Decimation Rate
CIC.N = 5; % Number of Stages
CIC.M = 1; % Differential Delay
CIC.IWL = 24; % Input Word Length
CIC.OWL = 24; % Output Word Length
Hcic = mfilt.cicdecim(CIC.R,CIC.M,CIC.N,CIC.IWL,CIC.OWL);
Hcicgain = dfilt.scalar(1/gain(Hcic)); % Define gain
Hcicnorm = cascade(Hcicgain,Hcic);

% CFIR Filter
CFIR.Passband_Edge = 38e3; % Passband edge (hz)
CFIR.Fs_in = CIC.Fs_in / CIC.R
CFIR.R = 1; % Decimation rate of CFIR
CFIR.N = 29; % Filter order
CFIR.Npow = CIC.N; % Sinc power
CFIR.w = 0.5; % Sinc frequency factor
CFIR.Apass = 5.7565e-03; % 0.01 dB
CFIR.Astop = 0.01; % 40 dB
CFIR.Aslope = 180; % 60 dB slope
CFIR.Fpass = CFIR.Passband_Edge / CFIR.Fs_in; % Passband-edge

cfir = firceqrip(CFIR.N,CFIR.Fpass,[CFIR.Apass, CFIR.Astop],...
    'passedge',...
    'slope',CFIR.Aslope,...
    'invsinc',[CFIR.w,CFIR.Npow]);
%Hcfir = mfilt.firdecim(CFIR.R,cfir);
Hcfir = dfilt.dfsymfir(cfir);
set(Hcfir,...
'Arithmetic', 'fixed',...
'CoeffWordLength', 24,...
'InputWordLength', 24);

% % PFIR
% PFIR.N = 29;
% PFIR.Fs_in = CFIR.Fs_in / CFIR.R
% PFIR.PassFreq = 20e3;
% PFIR.StopFreq = 24e3;
% PFIR.Fpass = PFIR.PassFreq/PFIR.Fs_in;
% PFIR.Fstop = PFIR.StopFreq/PFIR.Fs_in;
% PFIR.F = [0 PFIR.Fpass PFIR.Fstop 1];
% PFIR.A = [1 1 0 0];
% PFIR.W = [2 1]; % Weight the passband more than the stopband
% pfir = firgr(PFIR.N,PFIR.F,PFIR.A,PFIR.W);
% Hpfir = mfilt.firdecim(2,pfir);
% set(Hpfir,...
% 'Arithmetic', 'fixed',...
% 'CoeffWordLength', 16,...
% 'InputWordLength', 16);

Hcasc = cascade(Hcicnorm,Hcfir);%,Hpfir);
```

```
h = fvtool(Hcicnorm,Hcfir,Hcasc,'fs',[CIC.Fs_in CFIR.Fs_in CIC.Fs_in]);
%h = fvtool(Hcasc,'fs',CIC.Fs_in);
axis([0 .04 -20 12]);
legend(h,'CIC','CFIR','Cascaded Result','Location','NorthEast');
set(h,'ShowReference','off')
```

## *MATLAB Jitter Demonstration Code*

```
function jitter_demo();

tscale = 1e9;
jitter = 4e-9*tscale; % 4 ns
t_ideal = .498e-6*tscale;
f1 = 4e6/tscale; % 4 MHz - Over-sampling
f2 = 40e6/tscale; % 40 MHz - Under-sampling
full_scale = 2; % Peak-to-Peak max voltage

t=[0:.1:1000] .* 1e-9 * tscale;


% Jitter Sampling Position
t_jitter = t_ideal + jitter;

% Oversampling Case
y1 = sin(2*pi*f1.*t);

figure(1);
%subplot(2,1,1);
plot(t,y1,'LineWidth',1);

title('Effects of 10% Jitter - Over-Sampling','FontWeight','bold');
xlabel('Time (ns)');
ylabel('Amplitude (V)');
axis([.45e-6*tscale .55e-6*tscale -1 1]);
set(gca,'XTick',(0:1e-8:1e-6).*tscale);


% Ideal Cursors - Oversampling
% Add vertical cursor
ideal_vert_cursor_f1_x(1:2) = t_ideal;
ideal_vert_cursor_f1_y = [-1.2 sin(2*pi*f1*t_ideal)];
line(ideal_vert_cursor_f1_x,ideal_vert_cursor_f1_y,...
    'LineStyle','--',...
    'Color','k',...
    'LineWidth',2);

% Jitter Cursors - Oversampling
% Add vertical cursor
jitter_vert_cursor_f1_x(1:2) = t_jitter;
jitter_vert_cursor_f1_y = [-1.2 sin(2*pi*f1*t_jitter)];
line(jitter_vert_cursor_f1_x,jitter_vert_cursor_f1_y,...
    'LineStyle','--',...
    'Color','r',...
    'LineWidth',2);

legend('Signal','Ideal Sample','Jitter Sample');

% Ideal Cursors - Oversampling
```

```
% Add horizontal cursor
ideal_horiz_cursor_f1_x = [0 t_ideal];
ideal_horiz_cursor_f1_y(1:2) = sin(2*pi*f1*t_ideal);
line(ideal_horiz_cursor_f1_x,ideal_horiz_cursor_f1_y,...
    'LineStyle','--',...
    'Color','k',...
    'LineWidth',2);

% Jitter Cursors - Oversampling
% Add horizontal cursor
jitter_horiz_cursor_f1_x = [0 t_jitter];
jitter_horiz_cursor_f1_y(1:2) = sin(2*pi*f1*t_jitter);
line(jitter_horiz_cursor_f1_x,jitter_horiz_cursor_f1_y,...
    'LineStyle','--',...
    'Color','r',...
    'LineWidth',2);

amp_error_f1 = abs(sin(2*pi*f1*t_ideal) - sin(2*pi*f1*t_jitter));
amp_error_f1_fs = amp_error_f1 / full_scale;

rectangle('Position',[5.055e-7*tscale,.225,.42e-7*tscale,.15],...
    'FaceColor','w');
text(5.08e-7*tscale,.3, ['Amplitude Error: ',...
   num2str(amp_error_f1_fs*100,'%6.2f'),...
   '% FS']);

% Undersampling Case
y2 = sin(2*pi*f2.*t);

figure(2);
%subplot(2,1,2);
plot(t,y2,'LineWidth',1);

title('Effects of 10% Jitter - Under-Sampling','FontWeight','bold');
xlabel('Time (ns)');
ylabel('Amplitude (V)');
axis([.45e-6*tscale .55e-6*tscale -1 1]);
set(gca,'XTick',(0:1e-8:1e-6).*tscale);

% Ideal Cursors - Undersampling
% Add vertical cursor
ideal_vert_cursor_f2_x(1:2) = t_ideal;
ideal_vert_cursor_f2_y = [-1.2 sin(2*pi*f2*t_ideal)];
line(ideal_vert_cursor_f2_x,ideal_vert_cursor_f2_y,...
    'LineStyle','--',...
    'Color','k',...
    'LineWidth',2);

% Jitter Cursors - Undersampling
% Add vertical cursor
jitter_vert_cursor_f2_x(1:2) = t_jitter;
jitter_vert_cursor_f2_y = [-1.2 sin(2*pi*f2*t_jitter)];
line(jitter_vert_cursor_f2_x,jitter_vert_cursor_f2_y,...
    'LineStyle','--',...
    'Color','r',...
```

```
    'LineWidth',2);

legend('Signal','Ideal Sample','Jitter Sample');

% Ideal Cursors - Undersampling
% Add horizontal cursor
ideal_horiz_cursor_f2_x = [0 t_ideal];
ideal_horiz_cursor_f2_y(1:2) = sin(2*pi*f2*t_ideal);
line(ideal_horiz_cursor_f2_x,ideal_horiz_cursor_f2_y,...
    'LineStyle','--',...
    'Color','k',...
    'LineWidth',2);

% Jitter Cursors - Undersampling
% Add horizontal cursor
jitter_horiz_cursor_f2_x = [0 t_jitter];
jitter_horiz_cursor_f2_y(1:2) = sin(2*pi*f2*t_jitter);
line(jitter_horiz_cursor_f2_x,jitter_horiz_cursor_f2_y,...
    'LineStyle','--',...
    'Color','r',...
    'LineWidth',2);

amp_error_f2 = abs(sin(2*pi*f2*t_ideal) - sin(2*pi*f2*t_jitter));
amp_error_f2_fs = amp_error_f2 / full_scale;

rectangle('Position',[5.055e-7*tscale,.225,.42e-7*tscale,.15],...
    'FaceColor','w');
text(5.08e-7*tscale,.3, ['Amplitude Error: ',...
   num2str(amp_error_f2_fs*100,'%6.2f'),...
   '% FS']);
```

## *MATLAB Expected SNR Calculation Code*

```
function expected_signal()
% This script calculates the expected power for the signal and noise
% levels using the measured optical power from both the signal and lo
% branches of the optical setup

% 4 MHz Setup
T4.sig.opt = -13.6; % Optical power measured from signal branch (dBmW)
T4.lo.opt = -15; % Optical power measured from lo branch (dBmW)

T4.bandwidth = 14e6; % Filter bandwidth (MHz)
T4.preamp_noise_gain = 2.25e-12; % Pre-amp noise (A/rt(Hz))

% 40 MHz Setup
T40.sig.opt = -13.6; % Optical power measured from signal branch (dBmW)
T40.lo.opt = -14; % Optical power measured from lo branch (dBmW)

T40.bandwidth = 17e6; % Filter bandwidth (MHz)
T40.preamp_noise_gain = 15e-12; % Pre-amp noise (A/rt(Hz))

% General parameters
responsivity = 1; % Photodetector responsivity (A/W)
q = 1.6e-19; % Electron Charge (C)
Rin = 50; % Input impedance (ohms)
amp_gain = 40e3; % Gain of the amp (V/A)

% 4 MHz Case

T4.sig.opt = dbm2lin(T4.sig.opt); % Opt power from signal branch (W)
T4.lo.opt = dbm2lin(T4.lo.opt); % Opt power from lo branch (W)

T4.sig.i = responsivity * T4.sig.opt; % Current from signal branch (A)
T4.lo.i = responsivity * T4.lo.opt; % Current from lo branch (A)

T4.combined.i.dc = T4.sig.i + T4.lo.i; % Combined DC (A)
T4.combined.i.ac = 2 * sqrt(T4.sig.i * T4.lo.i); % Combined DC (A)

T4.combined.v.dc = amp_gain * T4.combined.i.dc; % Combined DC (V)
T4.combined.v.ac = amp_gain * T4.combined.i.ac; % Combined AC (Vp)
T4.combined.v.ac = T4.combined.v.ac / sqrt(2); % Combined AC (Vrms)

T4.combined.power.ac = (T4.combined.v.ac)^2 / Rin; % Comb AC power (W)
T4.combined.power.ac = lin2dbm(T4.combined.power.ac); % AC pwr (dBmW)

T4.preampnoise.v = amp_gain * T4.preamp_noise_gain * ...
sqrt(T4.bandwidth); % Preamp noise (V)
% Preamp noise pwr (W)
T4.preampnoise.power = (T4.preampnoise.v)^2 / Rin;
% Preamp noise power (dBmW)
T4.preampnoise.power = lin2dbm(T4.preampnoise.power);
T4.shotnoise.v = amp_gain * sqrt(2 * q * T4.combined.i.dc *
```

```
T4.bandwidth); % Shot noise (V)
T4.shotnoise.power = (T4.shotnoise.v)^2 / Rin; % Shot noise (W)
T4.shotnoise.power = lin2dbm(T4.shotnoise.power); % Shot noise (dBmW)

disp(['4 MHz Signal: ' num2str(T4.combined.power.ac) ' dBmW']);
disp(['4 MHz Shot Noise: ' num2str(T4.shotnoise.power) ' dBmW']);
disp(['4 MHz Preamp Noise: ' num2str(T4.preampnoise.power) ' dBmW']);

if T4.preampnoise.power > T4.shotnoise.power
    T4.snr = T4.combined.power.ac - T4.preampnoise.power;
    disp(['4 MHz SNR, Pre-amp noise limited: ' num2str(T4.snr) ' dB']);
else
    T4.snr = T4.combined.power.ac - T4.shotnoise.power;
    disp(['4 MHz SNR, Shot noise limited: ' num2str(T4.snr) ' dB']);
end;

% 40 MHz Case

T40.sig.opt = dbm2lin(T40.sig.opt); % Opt power from signal branch (W)
T40.lo.opt = dbm2lin(T40.lo.opt); % Opt power from lo branch (W)

T40.sig.i = responsivity * T40.sig.opt; % Current from signal branch
(A)
T40.lo.i = responsivity * T40.lo.opt; % Current from lo branch (A)

T40.combined.i.dc = T40.sig.i + T40.lo.i; % Combined DC (A)
T40.combined.i.ac = 2 * sqrt(T40.sig.i * T40.lo.i); % Combined DC (A)

T40.combined.v.dc = amp_gain * T40.combined.i.dc; % Combined DC (V)
T40.combined.v.ac = amp_gain * T40.combined.i.ac; % Combined AC (V)
T40.combined.v.ac = T40.combined.v.ac / sqrt(2); % Combined AC (Vrms)

% Comb AC power (W)
T40.combined.power.ac = (T40.combined.v.ac)^2 / Rin;
% Comb AC power (dBmW)
T40.combined.power.ac = lin2dbm(T40.combined.power.ac);
T40.preampnoise.v = amp_gain * T40.preamp_noise_gain *
sqrt(T40.bandwidth); % Preamp noise (V)
% Preamp noise power (W)
T40.preampnoise.power = (T40.preampnoise.v)^2 / Rin;
% Preamp noise power (dBmW)
T40.preampnoise.power = lin2dbm(T40.preampnoise.power);

T40.shotnoise.v = amp_gain * sqrt(2 * q * T40.combined.i.dc *
T40.bandwidth); % Shot noise (V)
T40.shotnoise.power = (T40.shotnoise.v)^2 / Rin; % Shot noise (W)
T40.shotnoise.power = lin2dbm(T40.shotnoise.power); % Shot noise (dBmW)

disp(['40 MHz Signal: ' num2str(T40.combined.power.ac) ' dBmW']);
disp(['40 MHz Shot Noise: ' num2str(T40.shotnoise.power) ' dBmW']);
disp(['40 MHz Preamp Noise: ' num2str(T40.preampnoise.power) ' dBmW']);

if T40.preampnoise.power > T40.shotnoise.power
    T40.snr = T40.combined.power.ac - T40.preampnoise.power;
    disp(['40 MHz SNR, Pre-amp noise limited: ' num2str(T40.snr) '
```

```
dB']);
else
    T40.snr = T40.combined.power.ac - T40.shotnoise.power;
    disp(['40 MHz SNR, Shot noise limited: ' num2str(T40.snr) ' dB']);
end;


function out = dbm2lin(in)
    out=(10^(in/10))/1000;

function out = lin2dbm(in)
    out=10*log10(in*1000);
```

# Appendix B – VHDL Code

## *Slave FPGA VHDL Code*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity main is
    Port (
               -- ADC Signals
               ADC_Clk_Out : out  STD_LOGIC;
               ADC_Clk_fb_I : in  STD_LOGIC;
               ADC_PU_n_I : in  STD_LOGIC;
               ADC_OTR_I : in  STD_LOGIC;
               ADC_LO_SIG_O : out  STD_LOGIC;
               ADC_ID_Bus_I : in  STD_LOGIC_VECTOR (2 downto 0);
               ADC_12Bus : in  STD_LOGIC_VECTOR (13 downto 0);
               ADC_34Bus : in  STD_LOGIC_VECTOR (13 downto 0);
               ADC_56Bus : in  STD_LOGIC_VECTOR (13 downto 0);
               ADC_78Bus : in  STD_LOGIC_VECTOR (13 downto 0);

               -- Clock Signals
               ClockIN1_14M : in  STD_LOGIC;
               ClockIN2_8M : in  STD_LOGIC;
               ClockIN3_33M : in  STD_LOGIC;
               ClockIN4_100M : in  STD_LOGIC;

               -- PCI Signals
               PCI_Clk_O : out  STD_LOGIC;
               PCI_Slv_Data_O : out STD_LOGIC_VECTOR (31 downto 0)
                              := x"00000000";
               PCI_Slv_DRdy_Bus_O : out STD_LOGIC);
end main;

architecture RTL of main is

      -- Component Declarations
      component ddc2r5mhz
            port (
            DIN: IN std_logic_VECTOR(11 downto 0);
            ND: IN std_logic;
            RDY: OUT std_logic;
            RFD: OUT std_logic;
            CLK: IN std_logic;
            DOUT_I: OUT std_logic_VECTOR(23 downto 0);
            DOUT_Q: OUT std_logic_VECTOR(23 downto 0));
      end component;

      component ddc4mhz
```

```
        port (
        DIN: IN std_logic_VECTOR(11 downto 0);
        ND: IN std_logic;
        RDY: OUT std_logic;
        RFD: OUT std_logic;
        CLK: IN std_logic;
        DOUT_I: OUT std_logic_VECTOR(23 downto 0);
        DOUT_Q: OUT std_logic_VECTOR(23 downto 0));
    end component;

    component BUFG
        port (I: in  STD_LOGIC;
                    O: out STD_LOGIC);
    end component;


    -- Clock Signals
    signal clk : std_logic;       -- 12.5 MHz
    signal clk180 : std_logic;    -- 12.5 MHz 180deg shifted clock
    signal clk180_nobuf : std_logic; --12.5 MHz 180deg w/o clk buffer

    signal sclk : std_logic;      -- Slow output clk
    signal sclk_nobuf : std_logic := '0';  -- Slow output clk w/o buf

    -- Slow clock counter

    signal sclk_count : std_logic_vector(7 downto 0) := x"00";

    constant DDC_R : integer range 1 to 512 := 256;
    constant num_of_DDCs : integer range 1 to 8 := 1;
    constant sclk_count_max : integer range 0 to 128
            := ((DDC_R / num_of_DDCs)/2)-1;

    -- DDC Signals
    type ddc_rec is record
        din : std_logic_vector(11 downto 0);
        din_buf : std_logic_vector(11 downto 0);
        nd : std_logic;
        rdy : std_logic;
        rfd : std_logic;
        dout_i : std_logic_vector(23 downto 0);
        dout_q : std_logic_vector(23 downto 0);
        dout_i_buf : std_logic_vector(23 downto 0);
        dout_q_buf : std_logic_vector(23 downto 0);
    end record;

    signal s_ddc0 : ddc_rec;
    signal s_ddc1 : ddc_rec;
    signal s_ddc2 : ddc_rec;
    signal s_ddc3 : ddc_rec;
    signal s_ddc4 : ddc_rec;
    signal s_ddc5 : ddc_rec;
    signal s_ddc6 : ddc_rec;
    signal s_ddc7 : ddc_rec;


    -- Synplicity black box declaration
```

```
      attribute syn_black_box : boolean;
      attribute syn_black_box of ddc2r5mhz: component is true;
      attribute syn_black_box of ddc4mhz: component is true;

      -- DDC Output State Machine Signals
      type state_type is (ddc0state,ddc1state,ddc2state,ddc3state,
                          ddc4state,ddc5state,ddc6state,ddc7state);
      signal state : state_type := ddc0state;

begin

      clk <= ADC_Clk_fb_I;    -- Set the primary system clock
      PCI_Clk_O <= clk;    -- Drive the PCI Bus with the system clock
      -- ADC Clock Pin, ADC is driven from external clock
      ADC_Clk_Out <= 'Z';


      -- Instantiate DDCs
      ddc0 : ddc2r5mhz
            port map (
                  DIN => s_ddc0.din,
                  ND => s_ddc0.nd,
                  RDY => s_ddc0.rdy,
                  RFD => s_ddc0.rfd,
                  CLK => clk,
                  DOUT_I => s_ddc0.dout_i_buf,
                  DOUT_Q => s_ddc0.dout_q_buf);

      ddc1 : ddc4mhz
        port map (
                  DIN => s_ddc1.din,
                  ND => s_ddc1.nd,
                  RDY => s_ddc1.rdy,
                  RFD => s_ddc1.rfd,
                  CLK => clk,
                  DOUT_I => s_ddc1.dout_i_buf,
                  DOUT_Q => s_ddc1.dout_q_buf);

      ddc2 : ddc2r5mhz
            port map (
                  DIN => s_ddc2.din,
                  ND => s_ddc2.nd,
                  RDY => s_ddc2.rdy,
                  RFD => s_ddc2.rfd,
                  CLK => clk,
                  DOUT_I => s_ddc2.dout_i_buf,
                  DOUT_Q => s_ddc2.dout_q_buf);

      ddc3 : ddc2r5mhz
            port map (
                  DIN => s_ddc3.din,
                  ND => s_ddc3.nd,
                  RDY => s_ddc3.rdy,
                  RFD => s_ddc3.rfd,
                  CLK => clk,
                  DOUT_I => s_ddc3.dout_i_buf,
                  DOUT_Q => s_ddc3.dout_q_buf);
```

```
ddc4 : ddc2r5mhz
      port map (
            DIN => s_ddc4.din,
            ND => s_ddc4.nd,
            RDY => s_ddc4.rdy,
            RFD => s_ddc4.rfd,
            CLK => clk,
            DOUT_I => s_ddc4.dout_i_buf,
            DOUT_Q => s_ddc4.dout_q_buf);

ddc5 : ddc2r5mhz
      port map (
            DIN => s_ddc5.din,
            ND => s_ddc5.nd,
            RDY => s_ddc5.rdy,
            RFD => s_ddc5.rfd,
            CLK => clk,
            DOUT_I => s_ddc5.dout_i_buf,
            DOUT_Q => s_ddc5.dout_q_buf);

ddc6 : ddc2r5mhz
      port map (
            DIN => s_ddc6.din,
            ND => s_ddc6.nd,
            RDY => s_ddc6.rdy,
            RFD => s_ddc6.rfd,
            CLK => clk,
            DOUT_I => s_ddc6.dout_i_buf,
            DOUT_Q => s_ddc6.dout_q_buf);

ddc7 : ddc2r5mhz
      port map (
            DIN => s_ddc7.din,
            ND => s_ddc7.nd,
            RDY => s_ddc7.rdy,
            RFD => s_ddc7.rfd,
            CLK => clk,
            DOUT_I => s_ddc7.dout_i_buf,
            DOUT_Q => s_ddc7.dout_q_buf);

BUFG_clk180 : BUFG
      port map (
            O => clk180,            -- Clock buffer output
            I => clk180_nobuf);  -- Clock buffer input

BUFG_sclk : BUFG
      port map (
            O => sclk,              -- Clock buffer output
            I => sclk_nobuf);       -- Clock buffer input


-- ADC to DDC routing
s_ddc0.nd <= '1';-- New data into the DDC every clock cycle
s_ddc1.nd <= '1';-- New data into the DDC every clock cycle
s_ddc2.nd <= '1';-- New data into the DDC every clock cycle
s_ddc3.nd <= '1';-- New data into the DDC every clock cycle
```

```
	s_ddc4.nd <= '1';-- New data into the DDC every clock cycle
	s_ddc5.nd <= '1';-- New data into the DDC every clock cycle
	s_ddc6.nd <= '1';-- New data into the DDC every clock cycle
	s_ddc7.nd <= '1';-- New data into the DDC every clock cycle

	ADC_LO_SIG_O <= '1'; -- Keep the AC signal feeding into the ADC

	-- Generate negative clock
	clk180_nobuf <= not clk;

	-- Generate slow clock for output data
	-- If CIC Decimation Rate = 32 and there are two DDCs, need to
	-- output data at 781.25 kHz or slow the 12.5 MHz clock by 16
	process(clk)
	begin
		if rising_edge(clk) then
			sclk_count <= sclk_count + 1;
			if (sclk_count = sclk_count_max) then
				sclk_nobuf <= not sclk_nobuf;
				sclk_count <= x"00";
			end if;
		end if;
	end process;

	-- Latch ADC Data Bus into appropriate DDC input buffers

	process(clk180)
	begin
		if rising_edge(clk180) then
			-- ADC Mux is high (Chan 0 to Data Bus A)
			s_ddc0.din_buf <= ADC_12Bus(13 downto 2);
			s_ddc2.din_buf <= ADC_34Bus(13 downto 2);
			s_ddc4.din_buf <= ADC_56Bus(13 downto 2);
			s_ddc6.din_buf <= ADC_78Bus(13 downto 2);
		end if;
	end process;

	process(clk)
	begin
		if rising_edge(clk) then
			-- ADC Mux is low (Chan 1 to Data Bus A)
			s_ddc1.din_buf <= ADC_12Bus(13 downto 2);
			s_ddc3.din_buf <= ADC_34Bus(13 downto 2);
			s_ddc5.din_buf <= ADC_56Bus(13 downto 2);
			s_ddc7.din_buf <= ADC_78Bus(13 downto 2);
		end if;
	end process;


	-- Latch DDC input buffers into DDC inputs

	process(clk)
	begin
		if rising_edge(clk) then
			s_ddc0.din <= s_ddc0.din_buf;
			s_ddc1.din <= s_ddc1.din_buf;
			s_ddc2.din <= s_ddc2.din_buf;
```

```
                s_ddc3.din <= s_ddc3.din_buf;
                s_ddc4.din <= s_ddc4.din_buf;
                s_ddc5.din <= s_ddc5.din_buf;
                s_ddc6.din <= s_ddc6.din_buf;
                s_ddc7.din <= s_ddc7.din_buf;
        end if;
    end process;

    -- Latch DDC output to output buffers

    process(clk)
    begin
        if rising_edge(clk) then
            if s_ddc0.rdy = '1' then
                s_ddc0.dout_i <= s_ddc0.dout_i_buf;
                s_ddc0.dout_q <= s_ddc0.dout_q_buf;
            end if;
        end if;
    end process;

    process(clk)
    begin
        if rising_edge(clk) then
            if s_ddc1.rdy = '1' then
                s_ddc1.dout_i <= s_ddc1.dout_i_buf;
                s_ddc1.dout_q <= s_ddc1.dout_q_buf;
            end if;
        end if;
    end process;

    -- Latch the DDC output buffers to the PCI Bus

    PCI_Slv_DRdy_Bus_O <= sclk;

    -- Channel Output Interleaver
    process(sclk)
    begin
        if rising_edge(sclk) then
            case state is

                when ddc0state =>
                    PCI_Slv_Data_O <= x"0" &
                                      s_ddc0.dout_i(13 downto 0) &
                                      s_ddc0.dout_q(13 downto 0);
                    state <= ddc1state;

                when ddc1state =>
                    PCI_Slv_Data_O <= x"1" &
                                      s_ddc1.dout_i(13 downto 0) &
                                      s_ddc1.dout_q(13 downto 0);
                    state <= ddc2state;

                when ddc2state =>
                    PCI_Slv_Data_O <= x"2" &
                                      s_ddc2.dout_i(13 downto 0) &
                                      s_ddc2.dout_q(13 downto 0);
                    state <= ddc3state;
```

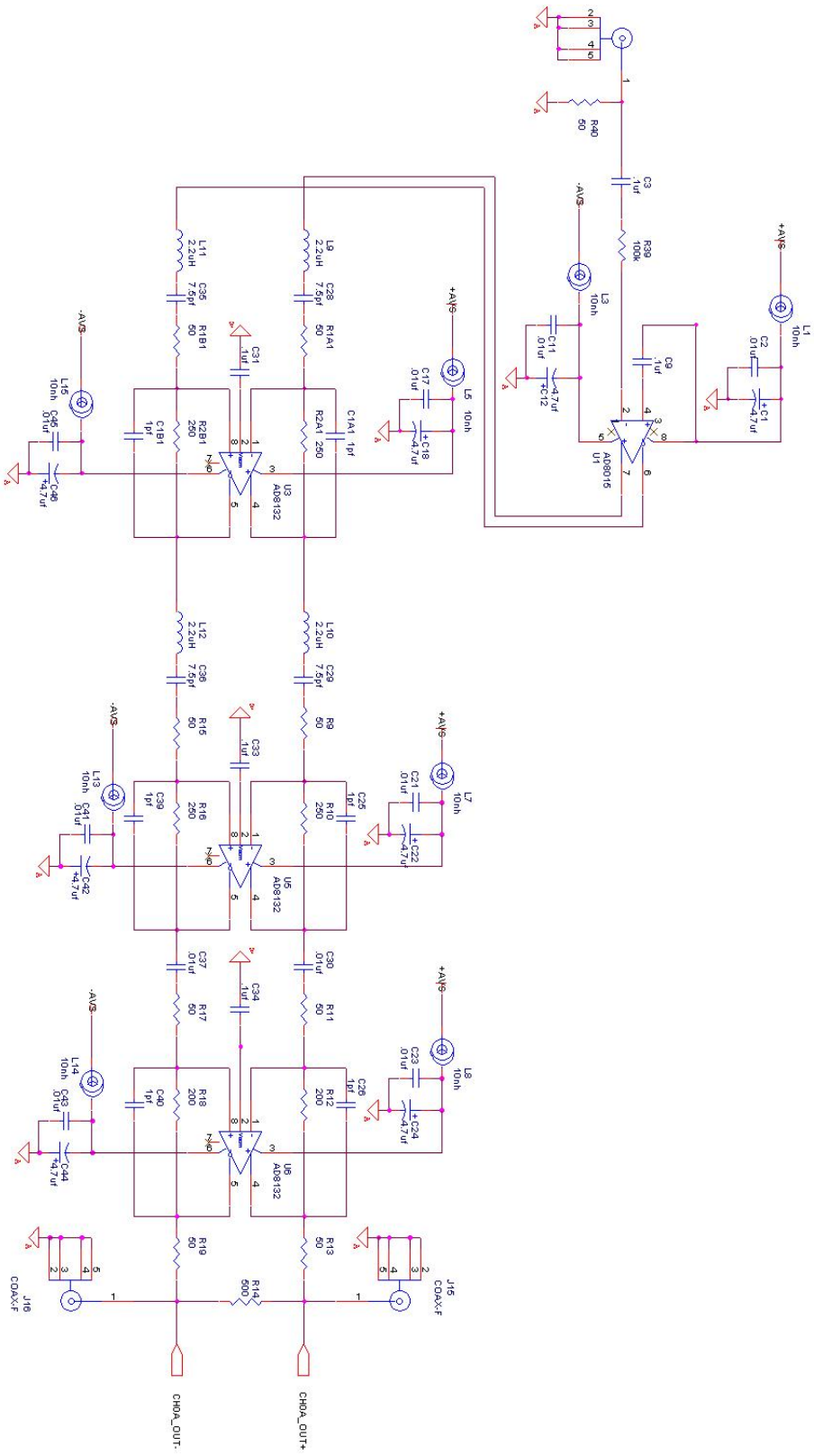
```
                        when ddc3state =>
                              PCI_Slv_Data_O <= x"3" &
                                          s_ddc3.dout_i(13 downto 0) &
                                          s_ddc3.dout_q(13 downto 0);
                              state <= ddc4state;

                        when ddc4state =>
                              PCI_Slv_Data_O <= x"4" &
                                          s_ddc4.dout_i(13 downto 0) &
                                          s_ddc4.dout_q(13 downto 0);
                              state <= ddc5state;

                        when ddc5state =>
                              PCI_Slv_Data_O <= x"5" &
                                          s_ddc5.dout_i(13 downto 0) &
                                          s_ddc5.dout_q(13 downto 0);
                              state <= ddc6state;

                        when ddc6state =>
                              PCI_Slv_Data_O <= x"6" &
                                          s_ddc6.dout_i(13 downto 0) &
                                          s_ddc6.dout_q(13 downto 0);
                              state <= ddc7state;

                        when ddc7state =>
                              PCI_Slv_Data_O <= x"7" &
                                          s_ddc7.dout_i(13 downto 0) &
                                          s_ddc7.dout_q(13 downto 0);
                              state <= ddc0state;

                        when others =>
                              state <= ddc0state;

                  end case;

            end if;
      end process;

end RTL;
```

## *Master FPGA VHDL Code*

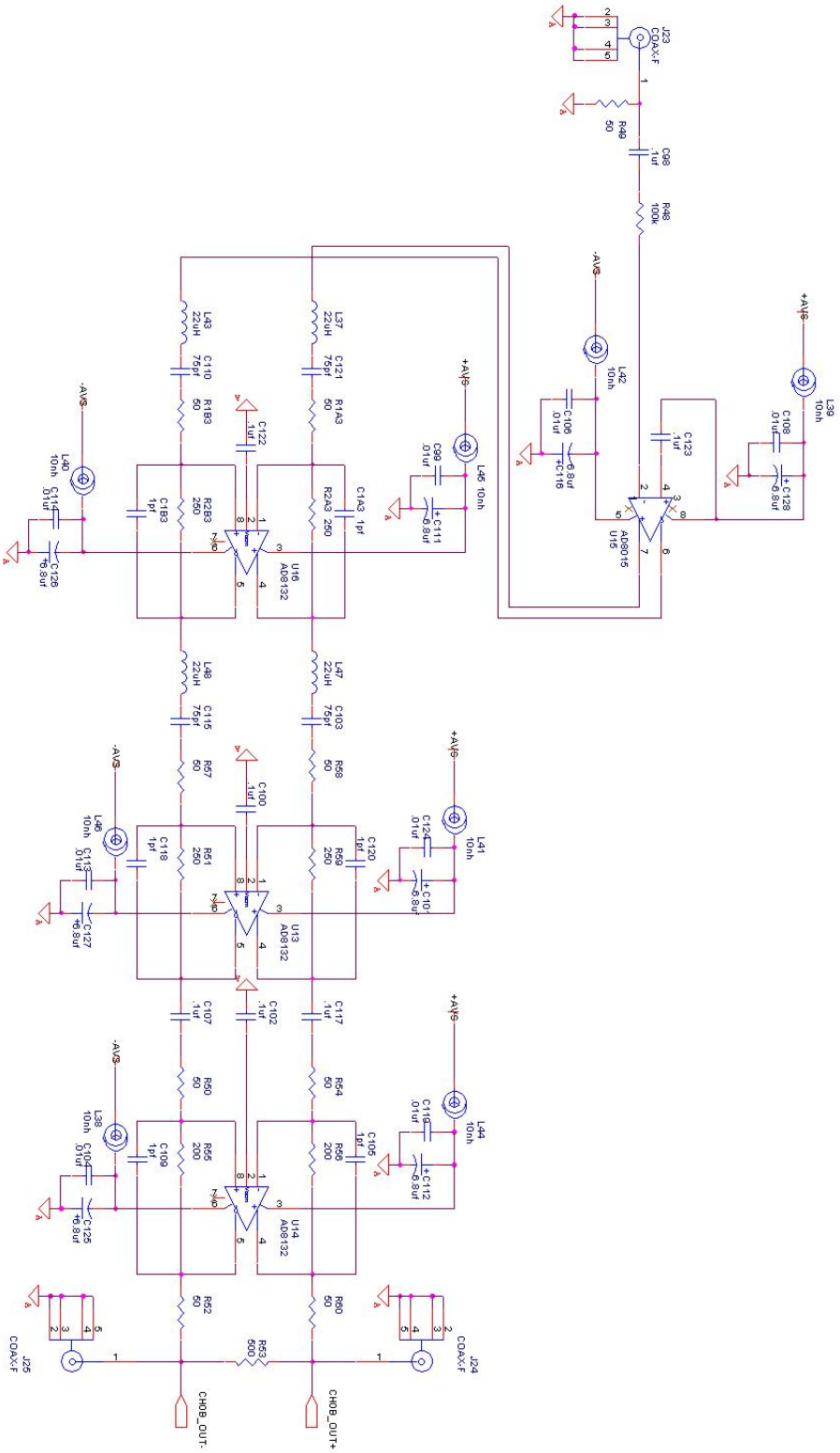
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity main_master is
    Port ( DataI : out  STD_LOGIC_VECTOR (13 downto 0);
                   DataQ : out  STD_LOGIC_VECTOR (13 downto 0);
                   ChanID : out STD_LOGIC_VECTOR (3 downto 0);
           PCI_CLK_O : in  STD_LOGIC;
           PCI_SLAVE_DATA_O : in  STD_LOGIC_VECTOR (31 downto 0);
                   PCI_Slv_DRdy_Bus_O : in STD_LOGIC;
                   GPIO_0 : out STD_LOGIC;
                   GPIO_1 : out STD_LOGIC);
end main_master;

architecture RTL of main_master is

      signal clk : std_logic;

begin

      clk <= PCI_CLK_O;
      GPIO_0 <= clk;
      GPIO_1 <= PCI_Slv_DRdy_Bus_O;

      process(clk)
      begin
            if rising_edge(clk) then
                  DataI <= PCI_SLAVE_DATA_O(27 downto 14);
                  DataQ <= PCI_SLAVE_DATA_O(13 downto 0);
                  ChanID <= PCI_SLAVE_DATA_O(31 downto 28);
            end if;
      end process;


end RTL;
```

# Appendix C – Schematics

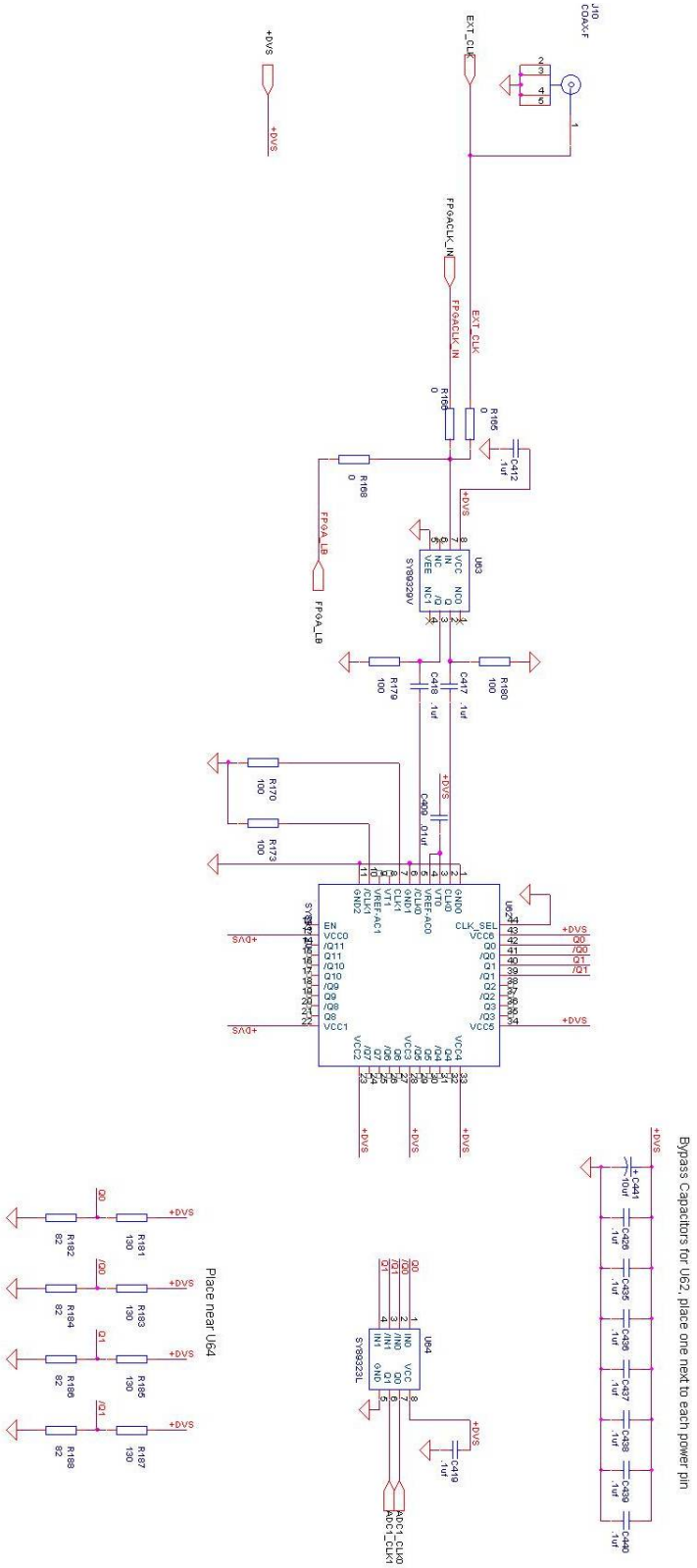
## *Under-Sampling Analog Circuit Schematic*



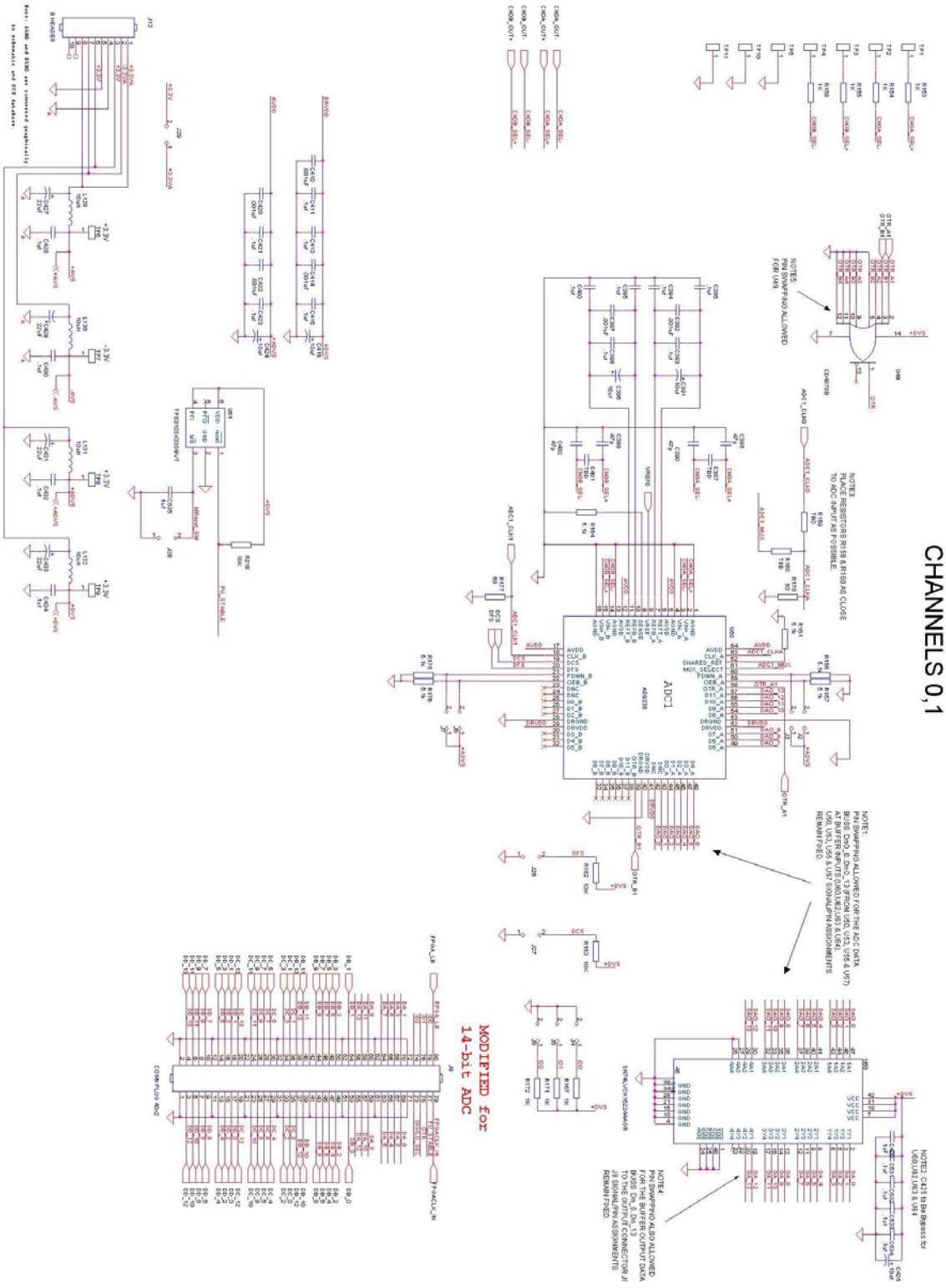
## *Over-Sampling Analog Circuit Schematic*



## *Clock Distribution Schematic*



## *ADC and Digital Circuit Schematic*



# Appendix D – ADC Evaluation System

## *Configuration*

Equipment:

- AD9238 Evaluation Board (Rev. D)
- HSC-ADC-EVAL-DC Capture Board (FIFO 3 Rev. A)
- ADC Analyzer 4.8.0
- Computer running Windows XP w/ parallel port
- Dual-Output Isolated Power Supply
- Function Generator for Clock Source

Configuring the computer and the software:

1. Set the parallel port to 'bidirectional' in the BIOS.
2. Install the ADC Analyzer Software.
3. Load the configuration file for the AD9238 Eval Board.
4. Turn off Channel A in the Buffer settings.
5. In the FFT settings for Channel B:
    a. Set the sampling rate to 12.5 MHz.
    b. Set the mode to 2's complement.

Configuring the AD9238 Evaluation Board:

1. Set the input to use the AD8138 differential amplifier by soldering a short on both JP15 and JP16.
2. Remove the RF transformer from the input by desoldering both JP13 and JP14.
3. Set a 1V internal reference by setting only JP2 of JP1 – JP5.
4. Connect a 3.3V power rail to AVDD, AVDD DUT, DRVDD DUT, and DVDD.
5. Connect a ground rail to the GND inputs.
6. Connect a -3.3V power rail to TP10.
7. To use the -3.3V power rail, set JP17 to position B.
8. Connect a 12.5 MHz sinusoidal source to the CLKA input.
9. Verify a square clock is generated by probing TP17.
10. Install JP29 (Mux Select).
11. Set JP22 to position A (Channel B).
12. Set JP25 to position A (Data Clock).
13. Remove JP24 (Separate clocks for each input channel).
14. Desolder JP6 and JP7 (Shared Reference).
15. Desolder JP27 (Not Mux Select).
16. Connect the input signal to BUFINA.

Configuring the Capture Board:

1. Connect a 3.3V power rail to VCC and DRVCC.
2. Set JP33 to down position (Dual Setting).
3. Remove JP34-41.
4. To set the mode to 2's complement, install JP12.
5. Rest of the jumpers should remain in the default position.

## *Implementation and Analysis*

Before designing the circuitry for the actual under-sampling vibrometry system, an evaluation board for the chosen ADC was obtained. The capture system, from Analog Devices includes an ADC Sampling board to digitize the analog signal and a high-speed FIFO Capture board to transmit the ADC data to a computer where it can be viewed with the ADC Analyzer software. Figure 67 shows a diagram of the system setup.

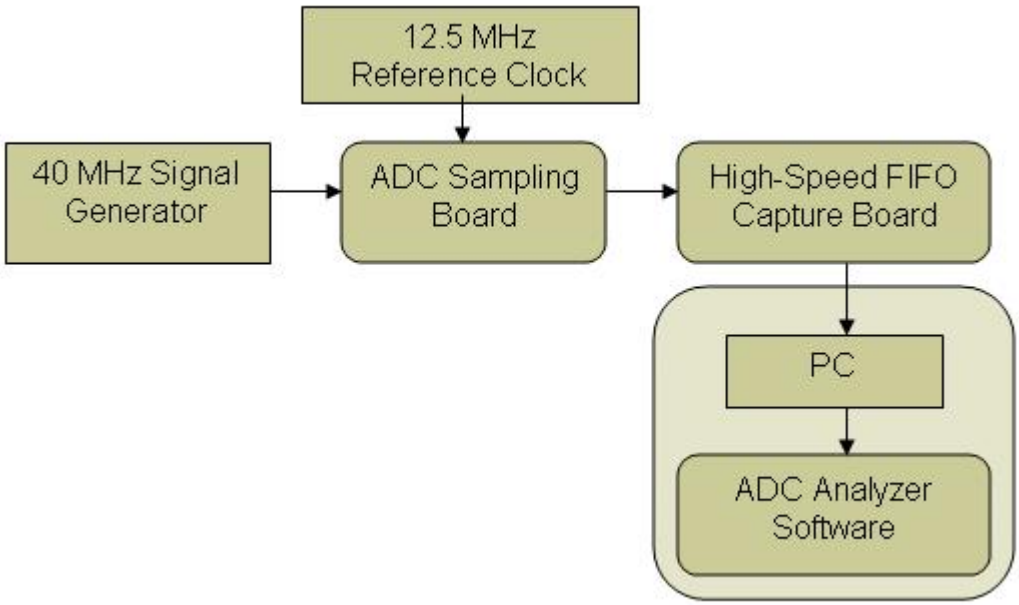



**Figure 67 - Block Diagram of ADC Evaluation System**

The 40 MHz input signal being generated is also phase-modulated by a 20 kHz signal to mimic the expected incoming light. A series of samples is then captured using the ADC Figure 68 shows the analyzer software and the waveform.

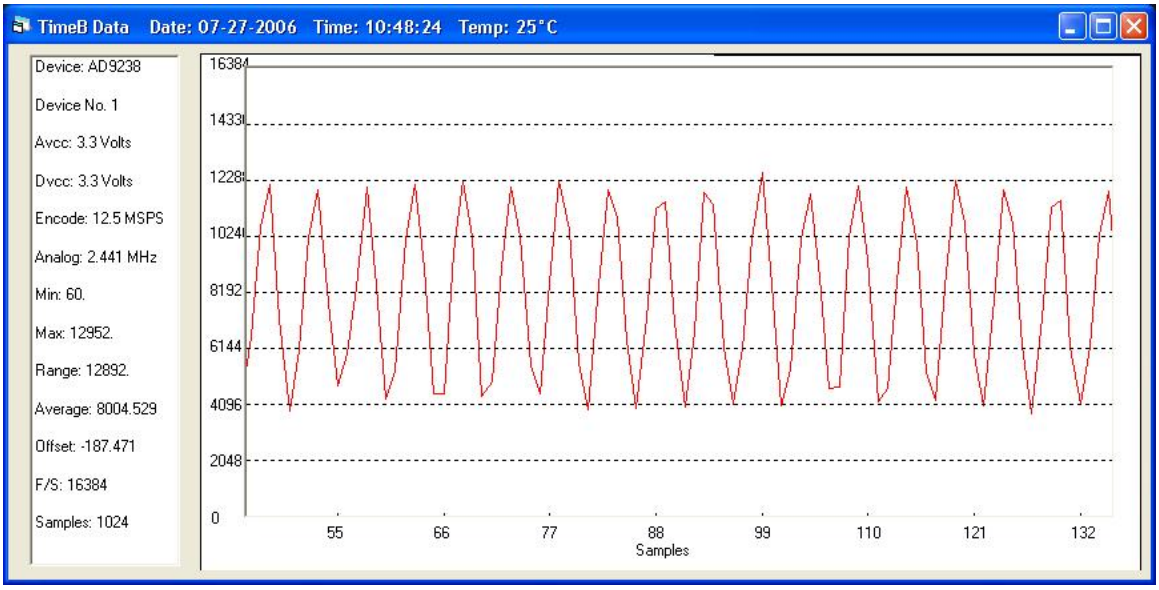



**Figure 68 - Waveform of Captured Data**

Figure 69 shows an FFT is then performed on the time samples to generate the frequency spectrum of the sampled data.

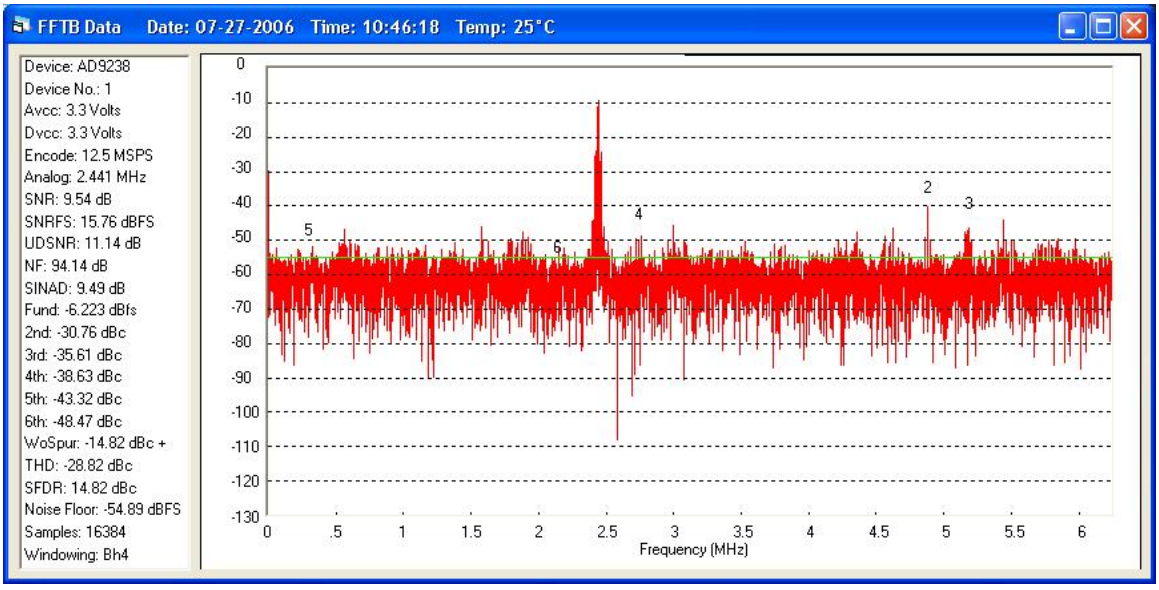



**Figure 69 - Spectrum of Captured Data**

As expected, the 40 MHz phase modulated signal reappears at 2.5 MHz. The performance of this captured data is expected to be less than that of the real system,

because of accuracy limitation in the function generators as well as the layout of the evaluation board used. The goal of this experiment was to verify that the under-sampling scheme modeled in Simulink works as expected.