Control of Microcantilevers Using a Stand-alone FPGA-Processor Platform

Solomon Davis

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2013

Reading Committee:

Joseph Garbini, Chair

John A. Sidles

Jonathan Jacky

Program Authorized to Offer Degree:
UW Department of Mechanical Engineering

University of Washington

Abstract

Control of Microcantilevers Using a Stand-alone FPGA-Processor Platform

Solomon Davis

Chair of the Supervisory Committee:
Professor Joseph Garbini
Department of Mechanical Engineering

In magnetic resonance force microscopy (MRFM), feedback control of the microcantilever is often needed to reduce the effective Q value. But because the natural frequency of the microcantilver is of order 10 KHz, this controller can be challenging to implement. Early MRFM experiments at the University of Washington used a field programmable gate array (FPGA) controller that worked well, but was expensive and time consuming to design, implement and maintain. However, there exist computing solutions that can perform the necessary functions of the earlier controller, for which program development and maintenance is less complex. This master's thesis explores the implementation of a single board platform (SBP) with a real-time processor and programmable FPGA to control the microcantilever. Two separate control methods were tested in this project: *conventional* and *heterodyne* control. The resulting data shows that this device is sufficient for control of the microcantilever when using either control method. Additionally, *heterodyne* control is advantageous, largely because it eliminates the need for many additional instruments, such as a lockin amplifier and spectrum analyzer. Therefore, with the combination of a SBP and the *heterodyne* control method, the separate instruments needed in our MRFM experiment have been reduced by half. Moreover, the cost of the SBP is a fraction of the combined cost of the equipment it has replaced. Finally, because SBPs are often programmed in common languages such as LabVIEW, the expertise required for program development and maintenance is substantially reduced.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

Magnetic Resonance Force Microscopy (MRFM) is a type of mechanical oscillator microscope (as opposed to optical or electron) with origins in magnetic resonance imaging (MRI) and atomic force microscopy. This device senses the magnetic forces exerted on it from a sample spin in order to perform extremely high resolution, nondestructive, three-dimensional imaging. This imaging technique holds promise for many fields of science, especially medicine [3].

On a basic level, the MRFM device consists of a micrometer scale cantilever with a ferromagnet at the tip, seen in Figure 1.1 [3]. To construct an image, a radio frequency magnetic field is applied to the sample, inducing spin resonance. These alternating magnetic moments create periodic forces that are detected by the microcantilever ferromagnet. The cantilever is moved over the sample and the ferromagnet polarizes the electron spins within the sample. The oscillations of the cantilever induced by the magnetic moments are detected by an optical interferometer. A deconvolution algorithm is used to reconstruct the three-dimensional molecular structure of the sample.

Because the forces induced by the applied magnetic field are so small, a cantilever with low damping is needed to achieve the sensitivity required. However, if the damping is low, the quality (Q value) will be large. With a more resonant cantilever comes a host of other issues such as large RMS amplitudes and longer ringing down times. Therefore, feedback control is required to reduce the effective Q without reducing the sensitivity of the microcantilever.

In selecting hardware to perform this control, a single-board platform (SBP) is attractive. One reason for this is that a SBP possesses three complementary resources: (1) a relatively low-speed real-time processor, capable of input/output and data presentation, (2) a FPGA

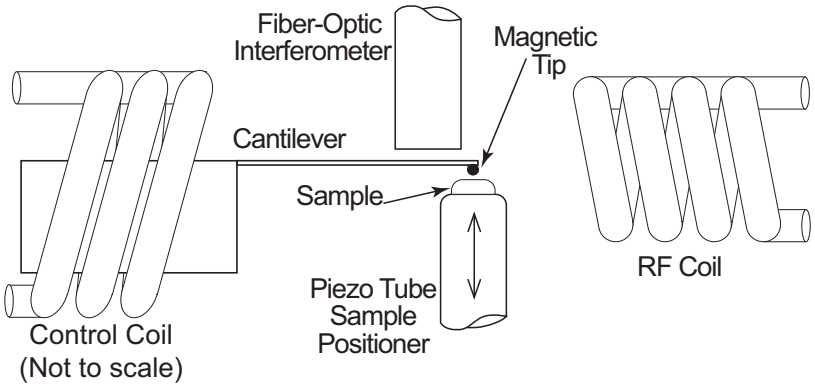

Figure 1.1: A schematic of a MRFM experiment

for continuous high-speed signal processing, and (3) analog and digital I/O channels. These integrated resources suffice to control the microcantilever, as well as perform additional simultaneous functions such as lockin filtering and spectral analysis. Furthermore, control and monitoring of all these functions can be done on a single computer screen. Moreover, the SBPs used in this project can be programmed in LabVIEW; a relatively easy language to learn and use. Therefore, the design, maintenance and use of the SBP and its algorithms are in many ways easier than that of control systems of similar complexity.

Two control methods were implemented and tested on the SBP: conventionally sampled control, and heterodyne control. Even though the conventional method worked well in the past, the performance of the heterodyne method is nearly identical. This method also possesses two computational advantages: (1) heterodyne filtering is automatically accomplished within the control algorithm, and (2) the phase-lead that can be added into the system is unlimited.

In Chapter 2, the mathematical models of the microcantilever as well as the conventional and heterodyne controllers are presented. In Chapter 3, the overall method for implementing and testing these control methods on a SBP is introduced. Additionally, the specific SBP model used in this project, as well as its capabilities and limitations are presented. Chapter 4 introduces the cantilever emulator, which was an essential tool for testing the control methods. The chapter then goes on to describe the development of the controllers in more

detail, and concludes with data demonstrating the effectiveness of controlling the cantilever emulator with the SBP. Finally, Chapter 5 presents data demonstrating the effectiveness of the SBP in control, spectral analysis and amplitude command on the cantilever emulator microcantilever.

Chapter 2

BACKGROUND

2.1 The Cantilever Model

The microcantilever can be modeled as a typical mass-spring-damper system via the second order differential equation

$$m\frac{d^2q(t)}{dt^2} + b\frac{dq(t)}{dt} + kq(t) = f(t)\,. \tag{2.1}$$

In this equation, m is the mass of the cantilever, b is the damping coefficient, k is the spring constant, *q(t)* is the position of the cantilever tip, and *f(t)* is the input force. In this model the higher order modes of the cantilever are ignored. The corresponding transfer function, relating the input force to the output displacement is

$$\frac{Q(s)}{F(s)} = \frac{1}{ms^2 + bs + k}\,.$$

The substitutions for the natural frequency $w_n = \sqrt{k/m}$ and the resonant quality $Q = \sqrt{km/b}$ are made.

$$G_{OL}(s) = \frac{Q(s)}{F(s)} = \frac{w_n^2/k}{s^2 + \frac{w_n}{Q}s + w_n^2} = \frac{1}{k(\frac{s^2}{w_n^2} + \frac{s}{Qw_n} + 1)} \tag{2.2}$$

In this form, the $1/k = 1/Q_c$ term is known as the resonance magnitude, and Q_c is the controlled quality.

There are two types of noise that affect the system. The first kind is thermal or process noise $w(t)$, which causes Brownian motion of the cantilever. The second kind is measurement noise $v(t)$ which arises from the limitations of the interferometer, filters and lockin amplifier. A diagram of the open loop system, including noise is shown in Figure 2.1. The equation for the entire open loop system in the frequency domain is

$$Q(s) = G_{OL}(s)[F(s) + W(s)] + V(s)\,. \tag{2.3}$$

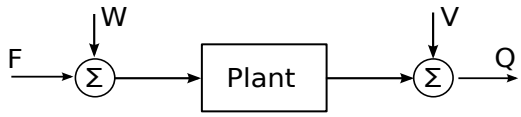

Figure 2.1: The open loop model with process and measurement noise

In this equation, $F(s)$, $W(s)$ and $V(s)$ are respectively the force input, process noise and measurement noise in the frequency domain. For $v(t) = \dot{q}(\mathrm{t})$ we have the following state space model:

$$\begin{bmatrix} \dot{q}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -w_n^2 & -\Gamma \end{bmatrix} \begin{bmatrix} q(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ w_n^2/k \end{bmatrix} \Big[f(t) + w(t) \Big] \,, \tag{2.4a}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} q(t) \\ v(t) \end{bmatrix} + v(t) \,. \tag{2.4b}$$

In Equations 2.4a and 2.4b, $\Gamma = w_n/Q$ is the full width at half maximum of the cantilever resonance peak, the cantilever bandwidth. Equations 2.4a and 2.4b are in the standard form

$$\dot{x}(t) = \boldsymbol{A}x + \boldsymbol{B}u + \boldsymbol{G}w \,,$$
$$y(t) = \boldsymbol{C}x + v \,.$$

2.2 Conventionally Sampled Control

The Brownian motion of the cantilever caused by the thermal noise is equivalent to motion caused by a white Langevin noise force with uniform power spectral density S_{ww} [1]. Therefore, the cantilever has average potential energy [2]

$$U_p = \frac{1}{2} k \langle q^2(t) \rangle = \frac{1}{2} k_B T \,, \tag{2.5}$$

where k_B is the Boltzman constant and T is the operating temperature. Rearranging Equation 2.5, the thermal noise contribution to the RMS amplitude of the uncontrolled cantilever is

$$\langle q^2(t) \rangle = \frac{k_B T}{k} \,. \tag{2.6}$$

It can be seen from Equation 2.6 that as k becomes small, the thermal noise contribution to the RMS amplitude becomes large. This has the effect of limiting the spacial resolution of the RMS device.

Furthermore, with a higher quality (per Section 1.1), three additional negative effects are produced. (1) Imaging data can only be collected within a certain bandwidth. As Q increases, the bandwidth becomes prohibitively small. (2) The ringing up time $\tau = 2Q/w_n = 2/\Gamma$. As Q increases, the ringing up time becomes large. (3) When k is small, there is greater potential for damage to the cantilever.

Garbini et al. [3] were among the first to explore the control of MRFM cantilevers in order to address the issues just mentioned. They demonstrated that the signal-to-noise ratio can never be improved through closed-loop control. Furthermore, they derived the optimal controller with LQR weighting matrices X=$\langle q^2(t)\rangle_{max}$, the maximum allowable mean-square variance in cantilever deflection, and U=$\langle u^2(t)\rangle_{max}$, the maximum allowable control force.

The optimal cantilever controller was found to be [3]:

$$
\begin{aligned}
H_{OC}(s) = \frac{U(s)}{Y(s)} &= \frac{K_{\mathrm{OC}}(s+z_{\mathrm{OC}})}{s^2+\omega_{\mathrm{OC}}/Q_{\mathrm{OC}}+\omega_{\mathrm{OC}}^2}\,, \\
\frac{K_{OC}}{w_n} &= \frac{1}{2}k\alpha\beta(\alpha+\beta)+\frac{k\alpha\beta}{Q}\,, \\
\frac{z_{\mathrm{OC}}}{w_n} &= \frac{\dfrac{\alpha\beta}{2}-2+\dfrac{\alpha+\beta}{Q}+\dfrac{2}{Q^2}}{\alpha+\beta+\dfrac{2}{Q}}\,, \\
\frac{\omega_{\mathrm{OC}}^2}{w_n} &= \frac{(\alpha+\beta)^2}{2}+\frac{\alpha+\beta}{Q}+1\,, \\
Q_{\mathrm{OC}} &= \frac{\sqrt{(\alpha+\beta)^2/2+(\alpha+\beta)/Q+1}}{\alpha+\beta+(1/Q)}\,,
\end{aligned}
\tag{2.7}
$$

where α is a measure of the control effort and β is a measure of the process noise. The equations defining α and β have been omitted here, but for this project α ranged from $0.001-0.01$ and β ranged from 0.01-0.02.

It can be easily shown that for a large Q and small values for α and β, a reasonable approximation of the optimal controller is [3]

$$
H_{OC}(s) = \frac{k\alpha\beta}{\dfrac{s^2}{w_n^2}+\dfrac{(\alpha+\beta)}{w_n}s+1}\,. \tag{2.8}
$$

2.3 Heterodyne Control

2.3.1 Heterodyning

First realized by Canadian engineer Reginald Fessenden, heterodyning is a signal processing technique where a new frequency (known as the intermediate frequency) is created by combining two frequencies. In 1901, Fessenden demonstrated how the heterodyne detector could make certain inaudible wave signals audible. The heterodyne detector was used in continuous wave transmitters to make morse code signals audible [4]. Heterodyning is used widely today in fields ranging from optics to musical synthesis.

There are two main elements of the heterodyning process. The first is called the downconversion, also known as "mixing-down," and the second is called the upconversion or "mixing-up".

In downconversion, the input signal is multiplied by a reference signal to create two channels with both high and low frequency components. Next, both channels are low-pass filtered. If the reference frequency is near the input frequency, then only near-DC signals will remain after filtering. These signals are known as the in-phase X and quadrature Y channels, and are the same X and Y channels produced by a lockin amplifier. At this stage, the input signal has been converted to the intermediate frequency (IF).

In the final step of the heterodyning process, mixing-up, the X and Y channels are again multiplied by reference sinusoids and then summed. If the frequency of the input and reference signals are the same, this final process constructs a signal with identical magnitude, frequency and phase as the input signal. These steps can be seen in Figure 2.2.

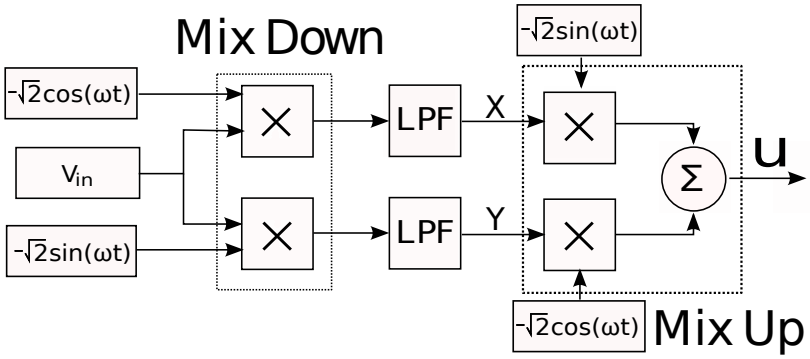

Figure 2.2: The heterodyne process

One immediate advantage of heterodyning is its ability to filter out noise. It can be seen in the demonstration below that all the components of the signal with frequencies other than the reference frequency are removed, which is useful for noisy systems. Previously, the MRFM experiment at the University of Washington used multiple lockin amplifiers for noise filtering; much of this filtering is now performed internally on the SBP.

There are two domains in which one can analyze the heterodyne process, the time domain and the frequency domain. The following demonstration, in the time domain, illustrates the noise filtering capabilities of heterodyning.

Assume the input signal *n(t)* is a sinusoid that consists of a signal component and a noise component. The signal component has frequency ω, phase θ and amplitude A. The noise has frequency θ_{n}, phase θ_{n} and amplitude A_{n}.

$$n(t) = A\sin(\omega t + \theta) + A_{\mathrm{n}}\sin(\theta_{\mathrm{n}} t + \theta_{\mathrm{n}})$$

The first step in the downconversion process is to multiply the input signal by the reference

sinusoid $\sqrt{2}\cos(\omega_{\text{ref}}\, t + \theta_{\text{ref}})$.

$$\begin{aligned} x_m(t) &= \left[A\sin(\omega t + \theta) + A_{\text{n}}\sin(\theta_{\text{n}} t + \theta_{\text{n}})\right] \times \sqrt{2}\cos(\omega_{\text{ref}}\, t + \theta_{\text{ref}})\,, \\ &= \frac{\sqrt{2}}{2} A \left(\sin\left(\left[\omega + \omega_{\text{ref}}\right] t + \left[\theta + \theta_{\text{ref}}\right]\right) + \sin\left(\left[\omega + \omega_{\text{ref}}\right] t - \left[\theta - \theta_{\text{ref}}\right]\right)\right) \\ &+ \frac{\sqrt{2}}{2} A_n \left(\sin\left(\left[\theta_{\text{n}} + \omega_{\text{ref}}\right] t + \left[\theta_{\text{n}} + \theta_{\text{ref}}\right]\right) + \sin\left(\left[\theta_{\text{n}} - \omega_{\text{ref}}\right] t + \left[\theta_{\text{n}} - \theta_{\text{ref}}\right]\right)\right)\,. \end{aligned}$$

We observe that if $\omega_{\text{ref}} \approx \omega$, x_m has a DC component. After low pass filtering, only the DC component remains. The X channel is

$$x(t) = \frac{\sqrt{2}}{2} A \sin\left(\left[\omega - \omega_{\text{ref}}\right] t + \left[\theta - \theta_{\text{ref}}\right]\right) = \frac{\sqrt{2}}{2} A \sin\left(\theta - \theta_{\text{ref}}\right)\,.$$

Similarly, to obtain the Y channel, we multiply the input signal by $-\sqrt{2}\sin(\omega_{\text{ref}}\, t + \theta_{\text{ref}})$. After low-pass filtering we are left with

$$y(t) = -\frac{\sqrt{2}}{2} A \cos\left(\left[\omega - \omega_{\text{ref}}\right] t + \left[\theta - \theta_{\text{ref}}\right]\right) = -\frac{\sqrt{2}}{2} A \cos\left(\theta - \theta_{\text{ref}}\right)\,.$$

At this point we observe that all noise components are absent.

In the mix-up, the two channels are again multiplied by reference sinusoids and then added to construct a signal identical to the signal component of the input, but which is without the noise component.

$$\begin{aligned} x(t) \times \sqrt{2} A\cos(\omega_{\text{ref}}\, t + \theta_{\text{ref}}) + y(t) \times\ -\sqrt{2} A\sin(\omega_{\text{ref}}\, t + \theta_{\text{ref}}) & \\ = A\sin(\theta - \theta_{\text{ref}})\cos(\omega_{\text{ref}}\, t + \theta_{\text{ref}}) + A\cos(\theta - \theta_{\text{ref}})\sin(\omega_{\text{ref}}\, t + \theta_{\text{ref}}) & \\ = A\sin(\omega_{\text{ref}}\, t + \theta) = A\sin(\omega\, t + \theta)\,, & \end{aligned}$$

This is the signal with no noise component.

2.3.2 Heterodyne Control

The first step in designing a heterodyne controller is to derive a model of the MRFM cantilever in terms of the IF. In order to do this, the cantilever is defined in terms of the states $x(t)$ and $y(t)$, and the "force signals" $g(t)$ and $h(t)$. In this section, the mix-up configuration (90 degree phase shift of the output) is used.

If the reference frequency equals that of the input signal, $\omega = \omega_{\text{ref}}$, then the linearized state space model of the cantilever in the IF is [1]

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} \frac{-\Gamma}{2} & 0 \\ 0 & \frac{-\Gamma}{2} \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} \frac{\omega_n}{2k} & 0 \\ 0 & \frac{\omega_n}{2k} \end{bmatrix} \begin{bmatrix} g(t) \\ h(t) \end{bmatrix}, \tag{2.9}$$

and the transfer function is found to be [1]

$$\begin{bmatrix} X(s) \\ Y(s) \end{bmatrix} = \frac{\omega_n}{2k\left(s^2 + \Gamma s + \left(\frac{\Gamma}{2}\right)^2\right)} \begin{bmatrix} s + \Gamma/2 & 0 \\ 0 & s + \Gamma/2 \end{bmatrix} \begin{bmatrix} G(s) \\ H(s) \end{bmatrix}. \tag{2.10}$$

For narrowband systems, the frequency response yields identical results to that of the plant represented in the more conventional form.

In Thomas E. Kriewall's 2004 Ph.D. dissertation [1], the optimal "zeroless" heterodyne controller is shown to be

$$\begin{bmatrix} G(s) \\ H(s) \end{bmatrix} = \begin{bmatrix} \frac{K_I}{s+\omega_I} & 0 \\ 0 & \frac{K_I}{s+\omega_I} \end{bmatrix} \begin{bmatrix} X(s) \\ Y(s) \end{bmatrix}, \tag{2.11}$$

$$\omega_I = \frac{\omega\left(1 + Q(\alpha+\beta)\right)}{2Q},$$

$$K_I = \frac{kQ\alpha\beta}{1 + Q(\alpha+\beta)}.$$

The optimal controller consists of two first order low-pass filters, each filtering an individual channel. These filters now serve two purposes: (1) Control of the X and Y channels and (2) filtering of the non-DC components required to perform the heterodyne process. The entire closed loop system can be seen in Figure 2.3.

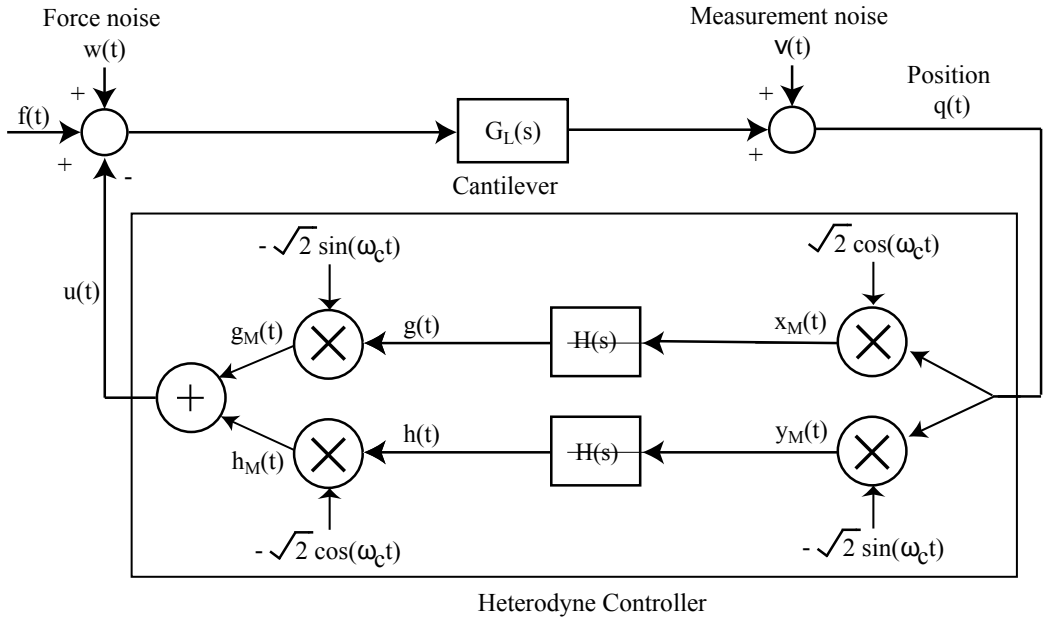

Figure 2.3: Closed loop system with hetrodyne controller, diagram adapted from [1]

Chapter 3

METHODOLOGY

3.1 Procedure

A typical MRFM experiment often includes a variety of interconnected instruments, performing the functions of signal processing, waveform generation and control. Clearly, much of this functionality could be performed on a SBP. Therefore, the goal of this project was not just to replace the custom controller, referred to as the Universal Serial Radio Peripheral (USRP) with a newer controller, but to replace much of the equipment with a SBP. This would reduce common grounding issues, maintenance effort and cost.

The following is a list of functions that could be performed on a SBP:

1) dynamic microcantilever control,
2) oscillation amplitude control,
3) noise filtering,
4) spectral analyzation of the interferometer signal, and
5) waveform generation.

The following is a list of equipment that would be replaced:

1) universal software radio peripheral (USRP),
2) two SR850 lockin amplifiers,
3) SR780 spectrum analyzer, and
4) BNC 555 pulse generator.

3.2 The NI Rio

3.2.1 Computing Levels of the Rio

The National Instruments Rio is a SBP with a conventional processor that communicates with an on-board FPGA, and is connected via ethernet to a host computer. This arrange-

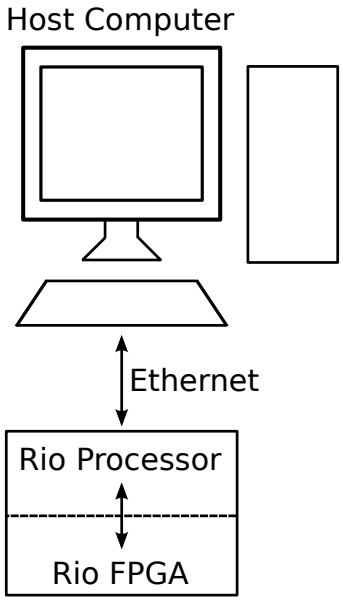

Figure 3.1: The three "levels" of the Rio

ment allows the functions required for the MRFM experiment to be distributed among the three levels of the SBP: host, processor and FPGA. These functions are distributed among the three levels according to their required computing intensity. The following is a proposed distribution of these functions:

Host: slow speed

- display/user input,
- file transfer, and
- communication with other experiment functions.

SBP processor: medium speed

- controller design,
- data processing, and
- host/FPGA communication.

FPGA: fast speed

- high speed sampling,
- lockin mix up/down, and
- control signal processing.

3.2.2 The 9632 sbRio Specifications

Figure 3.2: The National Instruments 9632 sbRio, picture from NI.com

The FPGA within the sb9632 is a Xilinx Spartan. This FPGA has a relatively small amount of memory and computing resources compared with the USRP as well the FPGAs installed on more recent models of the Rio. However, the Xilinx Spartan possesses the three powerful attributes of all FPGAs: reprogrammability, ability to process in parallel, and high processing speed.

The most basic unit of time of the Rio is known as a "tick". This is the unit used by the internal clock that runs at 40 MHz (25ns/tick). Every operation that is performed on the FPGA, for example multiplication or division, is completed within one tick.

On the FPGA, only fixed point math is allowed, though the word length and integer word length for every number and operation can be adjusted. This feature is useful because it facilitates fine-tuning of the FPGA code to optimize space and calculation time. The optimal word length is typically found through trial and error.

The processor on the Rio runs at 400 MHz. Because the FPGA is limited in space, many of the functions that do not need to be performed at a high rate can be placed on the processor and host levels. The FPGA and processor communicate with each other with first-in-first-out (FIFO) channels and what LabVIEW calls "controls" and "indicators". The host and the processor communicate through a 10/100BASE-T Ethernet cable.

The 9632 sbRio has 32 analog input channels with 16 bit resolution and a maximum voltage range of $\pm 10V$. The sampling time of these channels is approximately $4\mu s$ (250ks/s). These channels cannot sample simultaneously. The Rio also has 4 analog output channels with 16 bit resolution and a maximum voltage range of $\pm 10V$. The update time of these channels is approximately $3\mu s$. The Rio also has 110 bidirectional digital channels.

3.3 Method

This thesis covers the steps taken in designing, implementing and testing a SBP for MRFM control.

Chapter 4 demonstrates the need for microcantilever control in further detail. This chapter presents the method for determining the sample rate of the SBP under various conditions, and the results of these tests. It also presents the method for designing and implementing the cantilever emulator. This emulator is essential for testing the conventional and heterodyne controllers, because unlike the physical microcantilever, parameters of the emulator such as the Q value and the natural frequency can be varied. Next, the discrete *conventional* and *heterodyne* controller algorithms are shown. Finally, two additional functions performed by the SBP are presented; spectral analysis and amplitude command.

In Chapter 5, the resulting data from control of the emulator using either control method

is shown. This chapter also presents the control, spectral analysis and amplitude command data of the SBP with the physical microcantilever.

Chapter 4

IMPLEMENTATION

4.1 Filter Discretization

Before any continuous filter can be implemented, it must first be discretized. There are many methods for doing this, and in this project, Tustin's Method is used. This method is implemented by making the following substitution [5]

$$\frac{2}{T_s}\frac{(z-1)}{(z+1)} \to s\,, \tag{4.1}$$

or equivalently ,

$$G_d(z) = G(s)\big|_{s\to\frac{2}{T_s}\frac{(z-1)}{(z+1)}} = G\left(\frac{2}{T_s}\frac{(z-1)}{(z+1)}\right)\,, \tag{4.2}$$

where $T_s = \dfrac{1}{f_s}$ is the sample time of the computing device and f_s is the sample rate.

To determine the frequency response of the discrete transfer function, $G_d(z)$ is evaluated at $z = e^{jwT_s}$.

$$\begin{aligned} G_d(e^{jwT_s}) &= G\left(\frac{2}{T_s}\frac{e^{jwT_s}-1}{e^{jwT_s}+1}\right) \\ &= G\left(j\frac{2}{T_s}\tan\frac{wT_s}{2}\right) \end{aligned}$$

Necessarily, the frequency is effectively "warped" from w to $w_w = \dfrac{2}{T_s}\arctan\left(w\dfrac{T_s}{2}\right)$. To "prewarp" the natural frequency, Substitution 4.1 must be modified [5].

$$\frac{w}{\tan\left(\dfrac{wT}{2}\right)}\frac{z-1}{z+1} \to s \tag{4.3}$$

With this modification to Tustin's Method, the frequency response of the discrete transfer function matches that of the continuous at frequency w.

The discrete transfer function is rearranged into the following algorithm [5]

$$y[n] = \sum_{k=1}^{N}\frac{a_k}{a_0}y[n-k] + \sum_{k=0}^{M}\frac{b_k}{a_0}y[n-k]\,. \tag{4.4}$$

This is the form of a transfer function when implemented on a computing device.

4.2 Sample Rate

The digital implementation of a control system requires that one or more input channels be sampled, a control calculation be made, and one or more output channels updated. This cycle is repeated with period T_s. Moreover, it's useful to predict this sample rate under various computing conditions. In practice, the most time consuming functions performed on the FPGA level are the sampling and update times of the analog I/O channels. Therefore, the precise time consumed in updating these I/O channels should be determined.

To determine the average T_s for different combinations of input and output channels, each input channel is connected to an output channel, with no control calculation in between. The FPGA is timed while performing a given number of input-output iterations (i.e.100,000), where the input channels are sampled, and those values sent directly to one or many output channels. Data from these tests can be viewed in Table 4.1.

Table 4.1: Sample Data for the 9632 sbRio

Maximum Analog Sample Rates [kHz]					Maximum Analog Sample Times [μs]				
	0 in	1 in	2 in	3in		0 in	1 in	2 in	3 in
0 out	-	250	125	83.3	0 out	-	4	8	12
1 out	384.6	152.7	94.7	68.7	1 out	2.6	6.55	10.56	14.56
2 out	216.4	116.6	79.5	60.3	2 out	4.62	8.57	12.57	16.58
3 out	150.3	94.3	68.5	53.7	3 out	6.65	10.60	14.60	18.60

It can be seen in Table 4.1 that for every analog input channel, approximately $4\mu s$ per iteration is added. For every analog output channel, approximately $3\mu s$ per iteration is added. Therefore, the sample time can be rewritten as

$$T = 4\mu s \times (\text{number of analog input channels}) + 3\mu s \times (\text{number of analog output channels}) \ .$$

Two characteristics of the 9632 sbRio account for Equation 4.5 . (1) The analog-to-digital converter of the Rio cannot sample channels simultaneously. Therefore, if more then one analog input channel is used, those channels can only be sampled in sequence. (2) The Rio cannot be programmed to sample an input channel and update an output channel simultaneously. For this reason, the analog-to-digital converter must wait for the output channels to be updated before it can begin sampling again.

When using the Rio for sampling a high frequency signal, there are two main consequences of the limited input-output time (the total time taken by the I/O channels). The first is that there is the potential for under-sampling the signal. The second is T_s will never be zero. In other words, even if no calculation is performed, there will always be a phase-shift of the output waveform relative to the input waveform. This phase shift-becomes more pronounced as the waveform frequency becomes large.

The LabVIEW software allows the user to choose the sample rate of the Rio. Table 4.1 only indicates the limit to the sample rates. For a given channel combination, rates that are slower than what is displayed in the table can be chosen.

4.3 Phase-Lag

As mentioned in the previous section, any source of computing latency will result in a phase-shift of any internally generated sinusoids, or in the output of the controller itself. Therefore, it is important to understand the source of the latencies, and the impact they have on the system. It is also necessary to have a precise method for measuring the exact phase-lag, so that it can be corrected.

In the implementation of a SBP for control there are two sources of latency. The first is the conversion time of the I/O channels T_0, and the second is the calculation time T_C. The relationship between latency and phase-lag when using the Zero Order Hold method is [5]:

$$\phi = 1.5w(T_0 + T_C) \tag{4.5}$$

The effect of latency on the phase of the output can be seen in Figure 4.1.

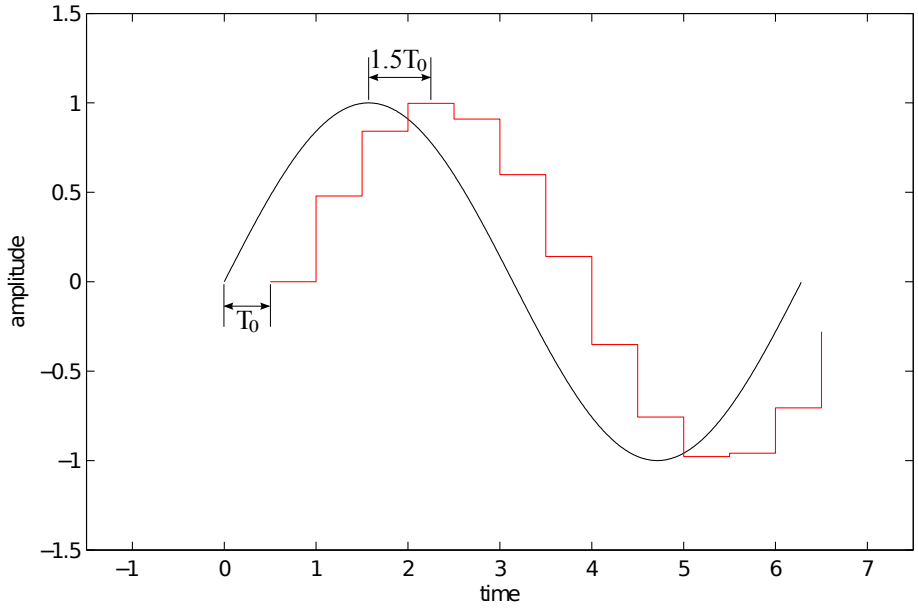

Figure 4.1: Phase-shift from the latency using the the Zero Order Hold method

To measure the phase-lag of the SBP, the coefficients of any filter algorithms must be chosen such that

$$\frac{X(s)}{Y(s)} = 1\,.$$

By doing this, the filter algorithms behave like a "wire" in that the input is equal to the output. Additionally, the calculation time T_C remains the same, as if the signal was being filtered. To accomplish this, all coefficients of a discrete transfer function in both the numerator and denominator must be zero except the last coefficients. These coefficients are given the value of 1. For example, if

$$\frac{X(s)}{Y(s)} = \frac{a_2 z^{-2} + a_1 z^{-1} + a_0}{b_2 z^{-2} + b_1 z^{-1} + b_0}\,,$$

$$\text{then } a_2 = a_1 = b_2 = b_1 = 0\,,$$

$$\text{and } a_0 = b_0 = 1\,.$$

A similar process must be done to the mix-up and mix-down of the heterodyne controller.

Finally, a sine wave is sent through the controller. This sine wave, and the output of

the controller are sent to a lockin amplifier. The lockin then compares the two waveforms and calculates the phase-shift ϕ_{Lag}.

The configuration for measuring the total phase-shift can be seen in Figure 4.2. To control the emulator properly, an equal amount of phase lead must be "put back" into the closed loop system via the controller.

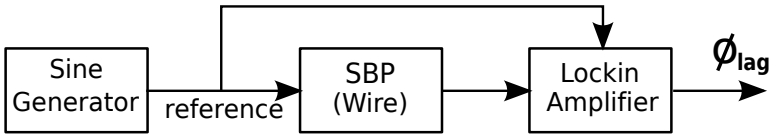

Figure 4.2: Configuration for measuring the total phase-lag from the SBP.

Additionally, in the MRFM experiment, there are two sources of phase-lag outside of the controller. The first is the latency from the interferometer T_{Int}, and the second is the latency from the amplification stages T_{Amp}. These sources can be seen in Figure 4.3

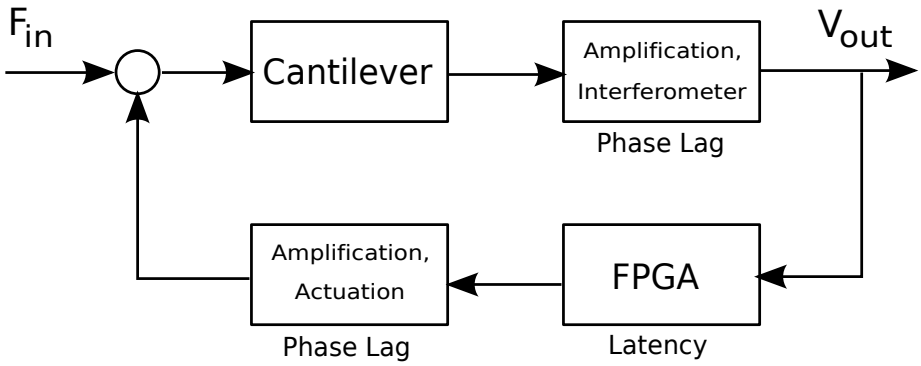

Figure 4.3: Sources of Phase-Lag in the MRFM Experiment

4.4 The Phase-Lead Compensated Controller

It can be seen in Figure 4.2, in the current state, the signal from the controller will lag by $(\phi_{Lag})_{tot} = \phi_{\text{Cont}} + \phi_{\text{Int}} + \phi_{\text{Amp}}$. This figure demonstrates the need for adding a phase-lead compensator into the closed loop system. The transfer function of the phase-lead

compensator and the controller is

$$H_{PL}(s) = H_{oc}(s) \times G_c(s)\,. \tag{4.6}$$

The compensator is of the form [6]

$$G_c(s) = \frac{(1+\eta\tau s)}{\sqrt{\eta}(1+\tau s)}\,, \tag{4.7}$$

where

$$\eta = \frac{1+\sin(\phi_m)}{1-\sin(\phi_m)}\,,$$

and

$$\tau = \frac{1}{\omega_m\sqrt{\eta}}\,.$$

In these equations, ϕ_m is the desired phase-lead at the center frequency w_m. For this project, the center frequency of the phase-lead compensator is chosen to be the natural frequency of the cantilever. This insures the best control at and around the natural frequency and is sufficient for a narrow-band system.

The entire transfer function of the compensated controller is [6]

$$\begin{aligned} H_{PL}(s) &= H_{oc}(s)G_c(s)\,, \\ &= \frac{K_{oc}}{K_a K_{pa} K_c K_i \sqrt{\eta}} \times \frac{(s+z_{oc})(1+\eta\tau s)}{(s^2+(\omega_{oc}/Q_{oc})s+\omega_{oc}^2)(1+\tau s)}\,. \end{aligned} \tag{4.8}$$

where K_i is the sensitivity of the interferometer, K_{pa} is the interferometer amplifier gain, K_a is the current amplifier gain, and K_a is the actuator gain.

Finally, the controller is discretized using the prewarped Tustin's approximation and

rearranged in the form of Equation 4.4. The coefficients are [6]

$$
\begin{aligned}
a_0 &= 1\,, \\
a_1 &= \frac{(-\omega' - 3\tau' + 3\omega'^3 + \omega'^2\tau')Q_{oc} + \omega'^2 - \omega'\tau'}{(\omega'^3 + \omega'^2\tau' + \omega'\tau')Q_{oc} + \omega'^2 + \omega'\tau'}\,, \\
a_2 &= \frac{(-\omega' + 3\tau' + 3\omega'^3 - \omega'^2\tau')Q_{oc} - \omega'^2 - \omega'\tau'}{(\omega'^3 + \omega'^2\tau' + \omega' + \tau')Q_{oc} + \omega'^2 + \omega'\tau'}\,, \\
a_3 &= \frac{(\omega' - \tau' + \omega'^3 - \omega'^2\tau')Q_{oc} - \omega'^2 + \omega'\tau'}{(\omega'^3 + \omega'^2\tau' + \omega' + \tau')Q_{oc} + \omega'^2 + \omega'\tau'}\,,
\end{aligned}
$$

and

$$
\begin{aligned}
b_0 &= \frac{Q_{oc}\omega' K'}{K_a K_{pa} K_c K_i \sqrt{\omega_n}} \times \frac{\omega' + \eta\tau' + \omega'\eta' z'_{oc}\tau' + \omega'^2 z'_{OC}}{(\omega'^3 + \omega'^2\tau' + \omega' + \tau')Q_{oc} + \omega'^2 + \omega'\tau'}\,, \\
b_1 &= b_0 \frac{\omega' - \eta\tau' + \omega'\eta z'_{oc}\tau + 3\omega'^2 z'_{oc}}{\omega'\eta\tau' + \omega'\eta z'_{OC}\tau' + \omega'^2 z'_{OC}}\,, \\
b_2 &= -b_0 \frac{\omega' + \eta\tau' + \omega'\eta z'_{oc}\tau' - 3\omega'^2 z'_{oc}}{\omega' + \eta\tau' + \omega'\eta z'_{oc}\tau' + \omega'^2 z'_{oc}}\,, \\
b_3 &= -b_0 \frac{-\omega' - \eta\tau' + \omega'\eta z'_{oc}\tau' - \omega'^2 z'_{oc}}{\omega' + \eta\tau' + \omega'\eta z'_{oc}\tau' + \omega'^2 z'_{oc}}\,,
\end{aligned}
$$

where $K' = \dfrac{K_{oc}}{\omega_{oc}}$, $\tau' = \dfrac{1}{\sqrt{\eta}}$, $z'_{oc} = \dfrac{z_{oc}}{\omega_{oc}}$, and $\omega' = \tan\dfrac{\omega_{oc}T}{2}$.

In Figures 7.3 and 7.4 in Appendix, the LabvVIEW front panel and block diagram of the conventional controller can be seen. This VI is on the FPGA and is not controlled by a processor level VI. For a given sample rate, ϕ_{lead}, Q, f_o and ω, the normalized filter coefficients and ticks per sample are calculated externally in Matlab and then loaded onto the FPGA VI directly.

4.5 The Heterodyne Controller: The FPGA Level

4.5.1 The Sine Generator

In section 1.4.1, the heterodyning process was discussed and the need for a sine wave demonstrated. If this waveform were to be generated externally (i.e. via a sine generator), then an additional analog input channel would be required. It can be seen in Table 4.1 that this additional channel would have a significant negative impact on the sample rate of the FPGA. Therefore, this sine wave is best generated internally on the FPGA.

Even though the memory on the FPGA is limited, the most efficient method for generating the sine wave is with a lookup table of pre-calculated points on a sinusoid. Each period of the generated sine wave is triggered by an external digital timing pulse with frequency f_{o}. This trigger insures that both the frequency and phase of the generated sine wave are in sync with other MRFM instrumentation external to the SBP.

Samples of the generated sine wave are constrained to being called at integer multiples of the internally generated clock. Therefore, the two parameters that necessarily need to be calculated to generate the waveform are: n, the number of samples per period, and m, the number of clock periods per sample.

There are two methods for calculating n and m. The first is referred to as the fixed sample rate (FSR) method, which maximizes n for a given clock frequency f_{c}. The algorithms for these parameters are

$$n_{\mathrm{FSR}} = \mathrm{floor}(f_{\mathrm{s_{max}}}/f_{\mathrm{o}}) + 1\,, \tag{4.9}$$

and

$$m_{\mathrm{FSR}} = \mathrm{floor}(f_{\mathrm{c}}/f_{\mathrm{s_{max}}})\,, \tag{4.10}$$

where $f_{\mathrm{s_{max}}}$ is the desired sampling rate of the FPGA. This integration of Equations 4.9 and 4.10 results in the following exact sample and sine wave frequencies.

$$f_{\mathrm{s_{final}}} = f_{\mathrm{c}}/m_{\mathrm{FSR}} \tag{4.11}$$

$$f_{\mathrm{o_{final}}} = \frac{f_{\mathrm{c}}}{n_{\mathrm{FSR}}\, m_{\mathrm{FSR}}} \tag{4.12}$$

The second method is referred to as the adjustable sample rate (ASR) method. This method calculates the parameters n and m such that the difference between the generated and nominal sine waves is minimized. The algorithms for parameters are

$$n_{\mathrm{ASR}} = \mathrm{floor}(f_{\mathrm{s_{max}}}/f_{\mathrm{o}}) + 1\,, \tag{4.13}$$

and

$$m_{\mathrm{ASR}} = \mathrm{floor}\left(\frac{f_{\mathrm{c}}}{n_{\mathrm{ASR}}\, f_{\mathrm{o}}}\right)\,. \tag{4.14}$$

Therefore the resulting exact sample and sine wave frequencies are

$$f_{\mathrm{s_{final}}} = f_{\mathrm{c}}/m_{\mathrm{ASR}} \tag{4.15}$$

and

$$f_{\mathrm{o_{final}}} = \frac{f_{\mathrm{c}}}{n_{\mathrm{ASR}}\, m_{\mathrm{ASR}}}\,. \tag{4.16}$$

In comparison with the FSR method, the ASR method sacrifices one (or more) sample points for a given waveform frequency to attain, on average, a smaller error between the generated and nominal waveforms. By reducing this error, the negatives effects from the spurious higher-order harmonics of the generated sine wave (such as noise peaks of the waveforms's power spectrum) are also reduced. The following figures demonstrate the performance of these two methods as the sine frequency f_{o} is varied. In these examples, the clock frequency f_{c} is 40 MHz, and the desired sampling rate $f_{\mathrm{s_{max}}}$ is 115 KHz.

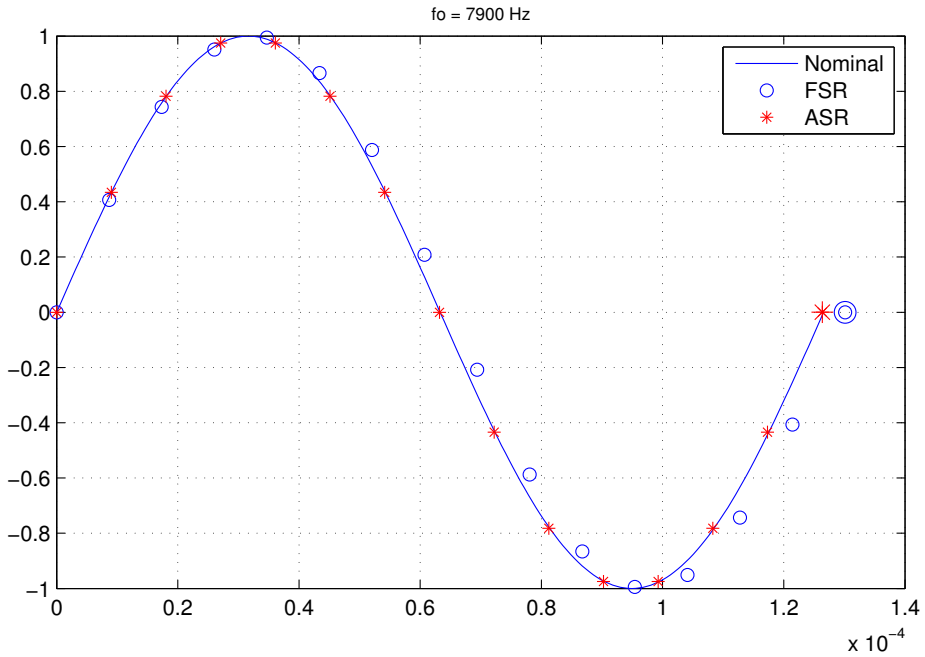

Figure 4.4: Comparison of the nominal sine wave with waveforms produced by the FSR and ASR methods, $f_o = 7900$ Hz

In Figure 4.4 it can be seen that the waveform produced by the FSR method has one more sample point than the waveform produced by the ASR method. However, the samples from the the FSR method don't match the nominal sinusoid as well as the ASR method. This is especially true for the final sample of the period. This discrepancy is referred to as the one-cycle endpoint error, and is characterized in Figure 4.7.

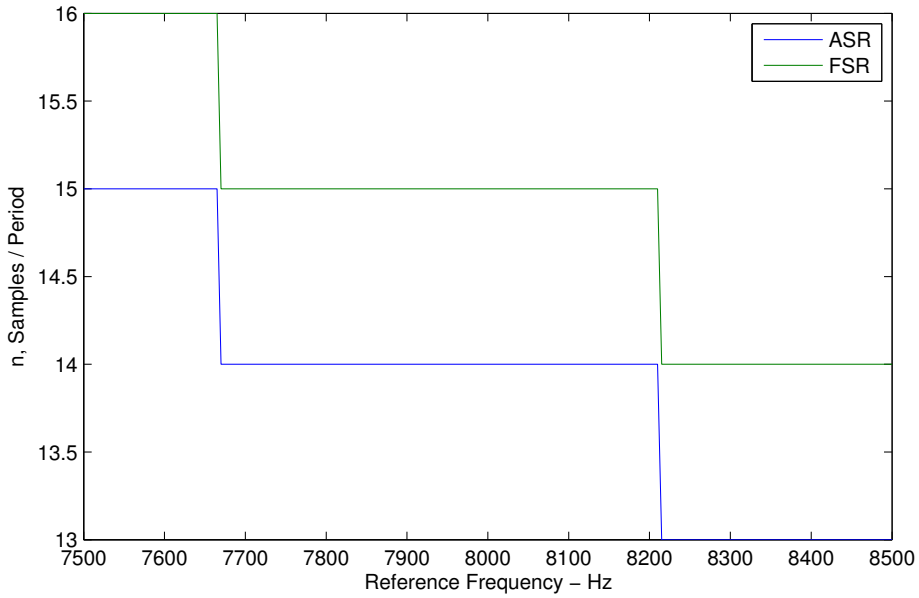

Figure 4.5: Comparison of the samples per period n of the ASR and FSR methods as the reference frequency f_o is varied

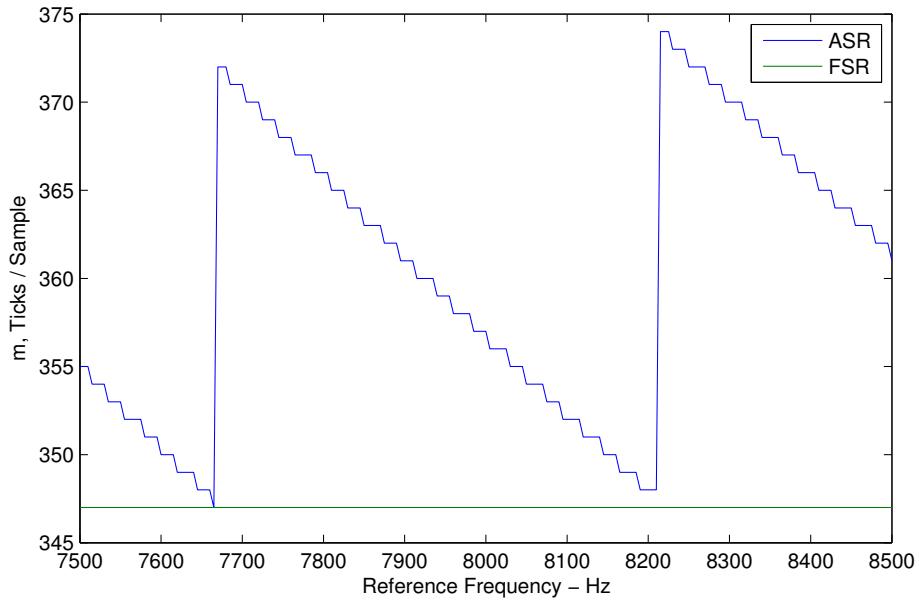

Figure 4.6: Comparison of the clock periods per sample m of the ASR and FSR methods as the reference frequency f_o is varied

Figures 4.5 and 4.6 demonstrate two important differences between the FSR and ASR methods. First, the FSR method will always generate a waveform with at least one more sample per period than the ASR method. Second, the ASR method adjusts the clock periods per sample as the reference frequency is varied, while the FSR method does not.

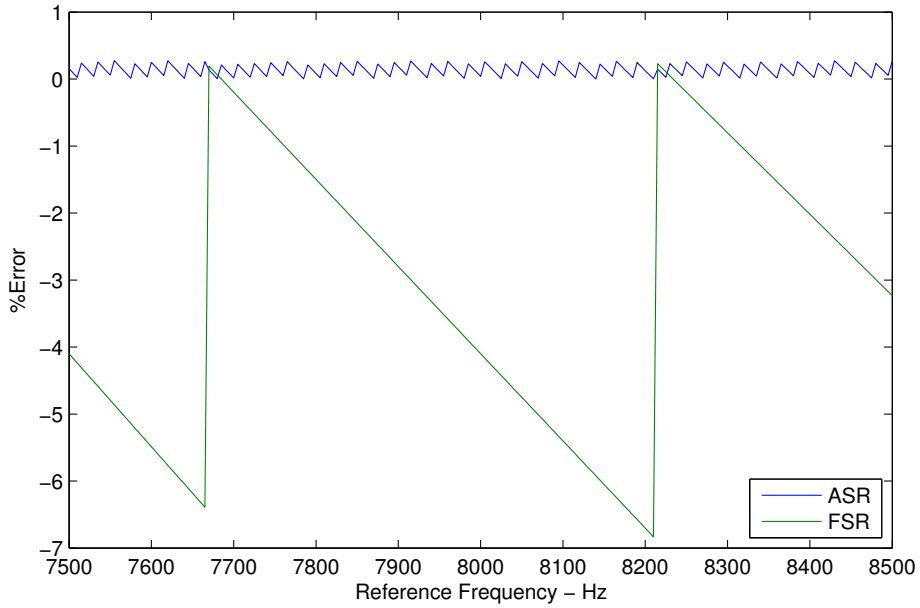

Figure 4.7: Comparison of the one-cycle endpoint error of the ASR and FSR methods as the reference frequency f_o is varied

In Figure 4.7, we see that the one-cycle endpoint error of both methods changes as the reference frequency is varied. However, from this figure it is clear that the error produced by the ASR method is almost always smaller (and in most cases, by a large margin) than the error produced by the FSR method. Therefore, it is the ASR method which is used for sine wave generation on the SBP.

4.5.2 The Phase Shift

Because T_0, T_c, T_{Int}, and T_{Amp} will never be zero, a phase-lag would occur at the output of the controller and cantilever emulator. For conventional control, this phase-lag is corrected at the natural frequency by inserting a phase-lead compensator G_c into the feedback

loop. However, with the heterodyne controller, phase-lead is added into the system quite differently. Instead of adding a compensator after the optimal filter, the phase is shifted by manipulating the X and Y channels within the controller itself.

Mathematically, the X and Y channels produced by heterodyning are the same as those produced by a lock-in amplifier. Therefore, the control signal can be plotted in the complex plane as a vector with amplitude $A = \sqrt{X^2 + Y^2}$ and phase $\phi = \arctan(Y/X)$. Therefore, all one needs to do to add phase into the control signal is to simply rotate the vector by the desired ϕ_{lead}. This can be seen in Figure 4.8.

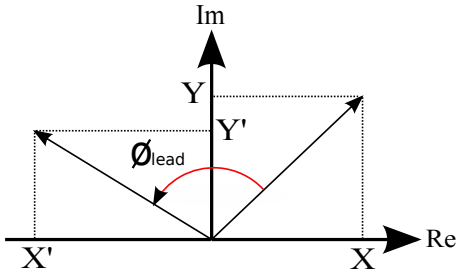

Figure 4.8: Adding phase lead into the system via X-Y vector rotation

To rotate the vector, both X and Y are multiplied by $e^{j\phi}$, and the complex and real components are separated.

$$\begin{aligned}(X + jY)e^{j\phi_{lead}} &= (X + jY)\left(\cos\phi_{lead} + j\sin\phi_{lead}\right) \\ &= (X\cos\phi_{lead} - Y\sin\phi_{lead}) + j(Y\cos\phi_{lead} + X\sin\phi_{lead})\end{aligned}$$

Therefore,

$$X' = X\cos\phi_{lead} - Y\sin\phi_{lead}\,, \tag{4.17a}$$

and

$$Y' = Y\cos\phi_{lead} + X\sin\phi_{lead}\,. \tag{4.17b}$$

The following is a list of the key characteristic of conventional and heterodyne phase lead.

Conventional Phase Lead:

- Adds exactly ϕ_{lead} of phase at only one frequency ω_c,
- influences the gain of the filter, and
- effectiveness has a limit of approximately 80 degrees.

Heterodyne Phase Lead:

- Adds exactly ϕ_{lead} of phase at every frequency,
- does not influence the gain of the filters, and
- effective for any value of ϕ_{lead}.

It is important to note that even though the phase-lead from either the compensator or the heterodyne controller correct the phase by the right amount at only one frequency w_c, the correction is sufficient for this application. This is because the microcantilever is a narrow-band system. Therefore, we only need to correct the phase at a single frequency, which for this application, is the natural frequency of the cantilever.

4.6 The Heterodyne Controller: The Processor Level

As the FPGA is performing its control algorithms, the X and Y values are periodically sent up to the processor through a FIFO. With this data, along with inputs from the user, the following key functions are performed on the processor VI.

- Decimation and Filtering: Even though all the X and Y data could used by the processor, in reality, most of this data is not needed to perform the necessary functions. Therefore these X and Y data are decimated, effectively allowing the processor to run at a faster rate. This decimation rate is specified by the user as the data sample frequency and is a function of the sample rate of the FPGA and the decimation number. Additionally, after this data is decimated, it is then filtered. The user can choose a filter with a roll-off of either 6 dB/Oct or 12 dB/Oct.

- Open-Loop Monitoring: Once the X and Y data is decimated and filtered, two important parameters are then calculated. These are the Vrms value of the interferometer, and the "noise temperature", which is an estimate of the "health" of the MRFM system. If this estimated temperature (calculated from equation 2.6) is close to the measured temperature, then it can be inferred that no excessive noise has entered the system. Additionally, three plots are displayed, and these are: Dual X and Y plots seen in Figure 4.9, the combined X and Y coordinates on the complex plane seen in Figure 4.10, and the power spectrum of the interferometer seen in Figure 4.11. The procedure for producing the power spectrum can be seen in the following Section 4.8.

- Closed-Loop Monitoring and Control: Just as in open-loop, the dual and complex plots of X and Y, as well as the power spectrum are displayed. Additionally, a user input allows the controlled quality Qc to be chosen. The available Qc ranges from 100 to 400 in increments of 50.

- Amplitude Command: When performing and MRFM experiment, the user may want to drive the cantilever at a certain amplitude to acquire phase-shift measurements. This amplitude can be specified through a user-input.

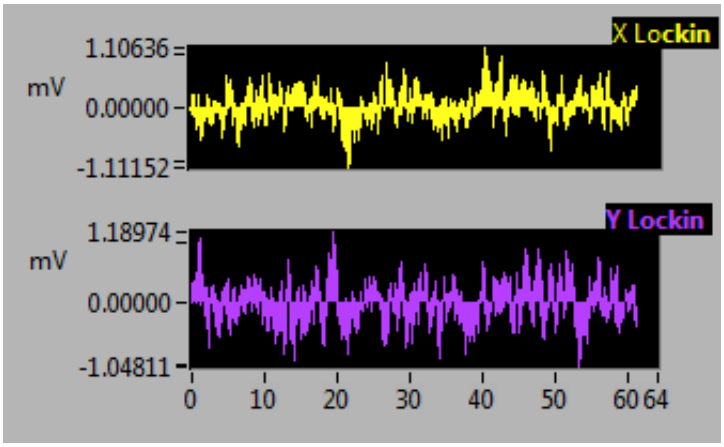

Figure 4.9: Front panel plots of the X and Y data from FPGA, transferred to the processor via FIFO

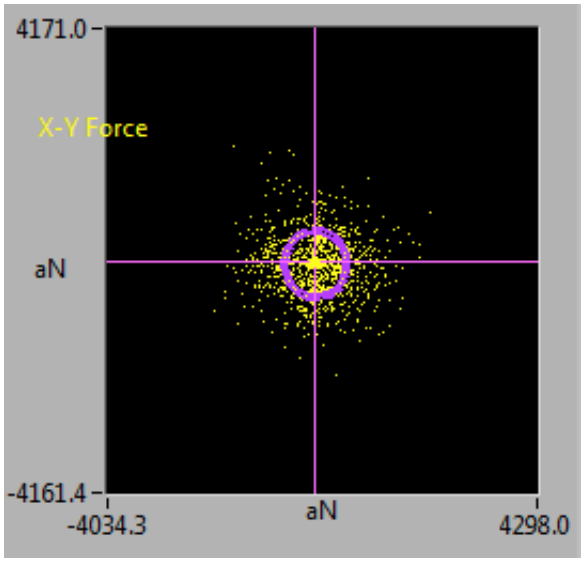

Figure 4.10: Front panel polar plot of the X and Y data from FPGA, transferred to the processor via FIFO

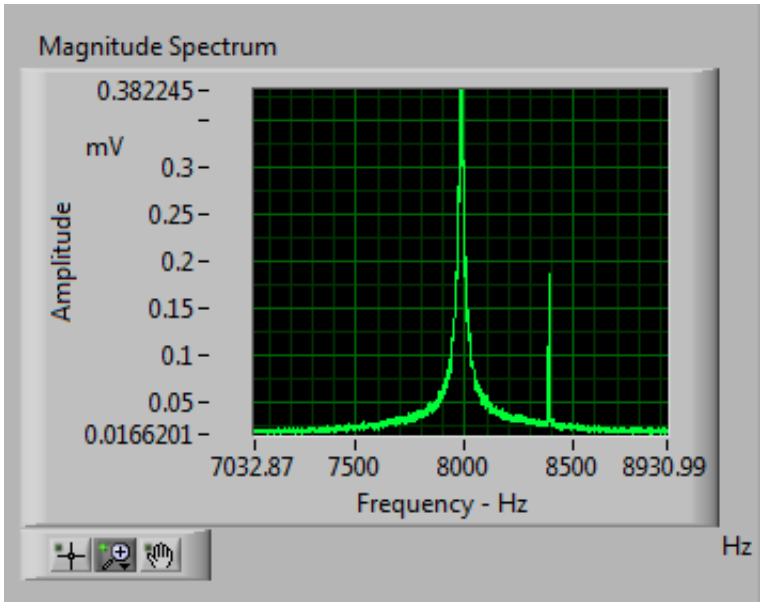

Figure 4.11: Front panel power spectrum of the X and Y data from the FPGA, transferred to the processor via FIFO

4.7 The Cantilever Emulator

As mentioned in Chapter 3, the cantilever emulator is used for testing the control of microcantilevers using a SBP. This emulator consists of the discretized model of the cantilever, implemented on the FPGA of a second SBP. In theory, cantilever emulator control is subject to the same voltage levels, A/D and D/A resolutions, and analog noises as the control of the physical cantilever (as seen in Figure 4.12). Moreover, the "physical" parameters of the emulator, such as its natural frequency, quality and spring constant can be varied. This is in contrast to control of the microcantilever, whose physical parameters cannot be varied. Therefore, the cantilever emulator should be useful for testing the performance of the controller under various conditions. This section covers, in detail, the design and implementation of the cantilever emulator.

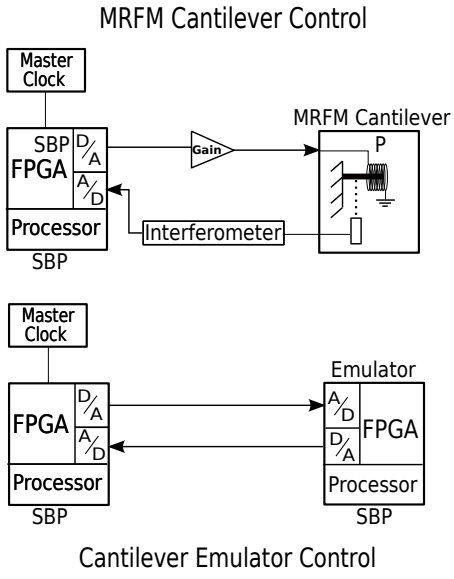

Figure 4.12: A comparison of the controlled microcantilever and controlled emulator with the SBP

The first step in constructing the cantilever emulator is to discretize the model of the microcantilever (per Section 2.1) with the method described in Section 4.1. The discretization of this model results in a biquad filter of the form

$$G_d(z) = \left. \frac{1}{Q_c(\frac{s^2}{w_{pw}^2} + \frac{s}{Qw_{pw}} + 1)} \right|_{s \to \frac{2}{T_s}\frac{(z-1)}{(z+1)}} , \tag{4.18}$$
$$= \frac{b_2 z^{-2} + b_1 z^{-1} + b_0}{a_2 z^{-2} + a_1 z^{-1} + a_0} ,$$

$$a_0 = Q_c \left(4 + \frac{2T_s\omega}{Q} + T_s^2\omega^2 \right) , \tag{4.19a}$$
$$a_1 = 2Q_c \left(-4 + T_s^2\omega^2 \right) , \tag{4.19b}$$
$$a_2 = Q_c \left(4 - \frac{2T_s\omega}{Q} + T_s^2\omega^2 \right) , \tag{4.19c}$$
$$b_0 = T_s^2\omega^2 , \tag{4.19d}$$
$$b_1 = 2T_s^2\omega^2 , \tag{4.19e}$$
$$b_2 = T_s^2\omega^2 , \tag{4.19f}$$

where T_s is the sampling time of the FPGA, ω is the natural frequency, and Q is the "native" quality of the emulator. Moreover, the overall scaling factor $1/Q_c$ in the emulator description establishes the appropriate signal amplitudes for the A/D and D/A conversion processes.

After discretization, the biquad filter is implemented on the SBP's FPGA. A diagram of the emulator algorithm can be seen in Figure 4.13. In this diagram, the first analog input channel receives the "force" signal, which in practice is produced by a sine generator. The second channel receives the control output from the first SBP. The analog output channel sends its signal to the input of the controller. The Front Panel and Block Diagram of the emulator can be seen in Figures 7.1 and 7.2 of the Appendix.

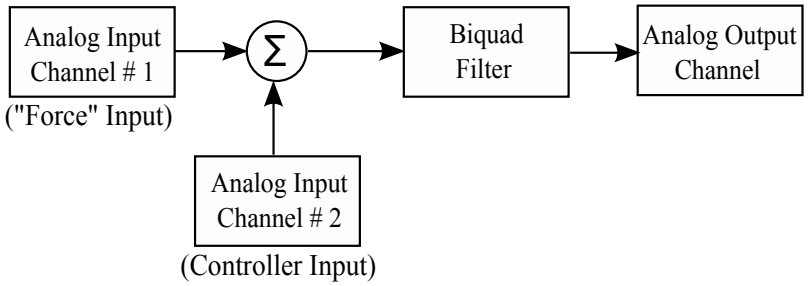

Figure 4.13: The cantilever emulator configuration on the SBP

4.8 Estimating the Power Spectral Density of a Narrow-Band Stationary Process from In-Phase and Quadrature Components

Assume $z(t)$ is a narrow-band stationary process with carrier frequency ω_0 where

$$z(t) = x(t)\cos(\omega_0 t) + y(t)\sin(\omega_0 t)\,, \tag{4.20}$$

where $x(t)$ and $y(t)$ are the in-phase and quadrature components. As in a lockin amplifier, the in-phase and quadrature components, and the central spectral tendency ω_0 are chosen to minimize the average rate of variation in $x(t)$ and $y(t)$. The Fourier Transform $\mathcal{F}$ of $z(t)$ is

$$Z(\omega) = \int_{-\infty}^{\infty} z(t)e^{-j\omega t}dt\,, \tag{4.21}$$

$$= \frac{1}{2}X(\omega-\omega_0) + \frac{1}{2}X(\omega-\omega_0))\,, \tag{4.22}$$

$$- \frac{1}{2j}Y(\omega-\omega_0) + \frac{1}{2j}Y(\omega-\omega_0)\,, \tag{4.23}$$

where $X(\omega) = \mathcal{F}(x(t))$, and $Y(\omega) = \mathcal{F}(x(t))$.

Converting to a discrete system, we must approximate the continuous Fourier Transforms $X(\omega)$ and $Y(\omega)$ by the Discrete Fourier Transform (DFT). For example

$$X_k = \sum_{i=0}^{N-1} x_k e^{-j2\pi k/N}\,, \qquad k = 0, 1, 2, ..., N_1\,. \tag{4.24}$$

We can now use X_k and Y_k to compute the approximate $Z(\omega)$ in Equation 4.23. And finally, our estimate of the power spectral density of $z(t)$ is [7]

$$\tilde{G_k} = \frac{2\Delta t}{N}|Z_k|^2\,. \tag{4.25}$$

In practice, this gives a power spectral density estimate for one set of data. In this project however, as sets of data continue to be sent up from the FPGA to the processor, the power spectral density estimate is ensemble averaged to obtain a smoother, and more accurate spectral density.

4.9 Commanded Amplitude

As mentioned in Section 4.6, an important feature of the SBP is the ability to set the amplitude of the microcantilever. This "Commanded Amplitude" is referred to as X_{CMD}, and can be seen in functional block diagram of Figure 4.14.

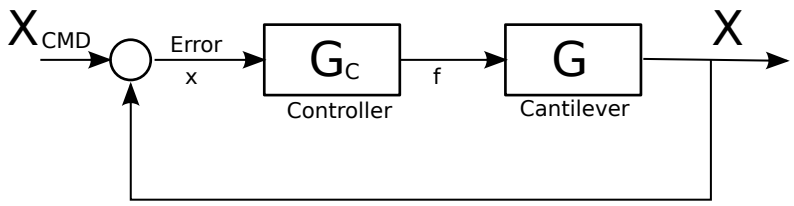

Figure 4.14: Block digram of the feedback loop

In determining the effectiveness of commanding the microcantilever amplitude, two important data sets can be looked at; the measured output amplitude, and the deviation of the output amplitude from the commanded amplitude, also known as the error signal. Referring to Figure 4.14, the following is a derivation of the error signal [6]:

$$\frac{Error}{C_{\mathrm{CMD}}} = \frac{1}{1 + G_{\mathrm{C}} G}\,, \tag{4.26a}$$

where

$$G_{\mathrm{C}} = \frac{K}{\tau s + 1}\,, \tag{4.26b}$$

$$G = \frac{Q/K}{1 + \dfrac{2Q}{\omega} s}\,, \tag{4.26c}$$

$$K = \frac{k\alpha\beta}{1 + Q(\alpha + \beta)}\,, \tag{4.26d}$$

$$\tau = \frac{2}{\omega(\alpha + \beta)}\,, \tag{4.26e}$$

$$\alpha = \frac{1}{Q_c}\,. \tag{4.26f}$$

If $s \to j\omega$ and $j\omega \to 0$ then

$$\frac{Error}{X_{\mathrm{CMD}}} = \frac{1 + Q(\alpha + \beta)}{(1 + Q\alpha)(1 + Q\beta)}\,. \tag{4.27}$$

This is the transfer function of the error signal with respect to the commanded amplitude. The commanded amplitude is implemented by summing V_{ref} and the X channel before the low-pass filter, where

$$V_{\mathrm{ref}} = \frac{X_{\mathrm{amp}}\, K_{\mathrm{i}}}{\sqrt{2}}\,, \tag{4.28}$$

and where X_{amp} is the desired amplitude of the microcantilever. This summation can be seen in Figure 4.15.

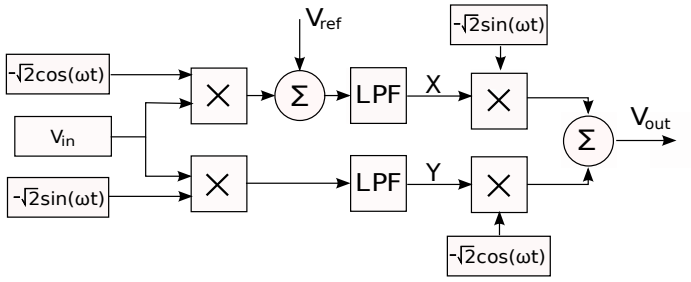

Figure 4.15: Implementation of the commanded amplitude by summing V_{ref} and the X channel

Chapter 5

EXPERIMENTAL RESULTS

5.1 The Cantilever Emulator

In Section 4.7, the cantilever emulator was discussed as a method for testing the SBP controller (conventional and heterodyne) under a range of parameters. Prior to these tests however, a frequency sweep of the emulator by itself (uncontrolled) was performed. This was to determine if the SBP really does emulate the microcantilever accurately. Figure 5.1 presents the data from this frequency sweep, superimposed with the bode plot of the cantilever model (Equation 2.2).

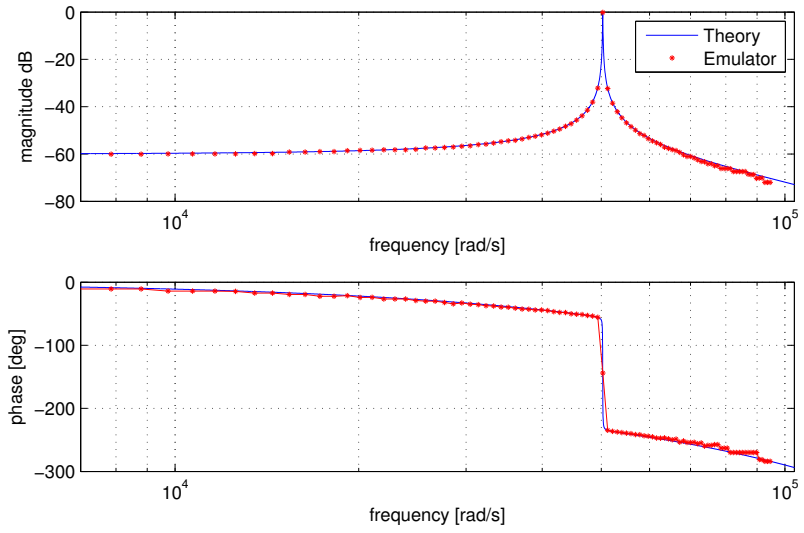

Figure 5.1: A comparison Frequency sweep of the cantilever emulator with the theory. Q = 1000, k = 1000 and f_o = 8000

It can be seen in this plot that the cantilever emulator was successful in mimicking the physical cantilever, and was therefore used in testing the controller.

Since it had been determined that the cantilever emulator mimics the dynamic behavior

of the microcantilever accurately, the performance of the SBP controller was tested under varying conditions, particularly with a varying spring constant of the cantilever emulator. Like the data in Figure 5.1, the tests consisted of a frequency sweep, only this time the emulator was controlled by the SBP controller.

The following figures are magnitude and phase plots comparing: the discretized controlled-mirocantilever model, the controlled cantilever emulator using the heterodyne control method and the controlled cantilever emulator using the conventionally sampled control method. Each plot represents a different test, where the spring constant and controlled quality are changed incrementally. Moreover, the natural frequency of and the native quality of the emulator were set at 8000 Hz and 1000 respectively.

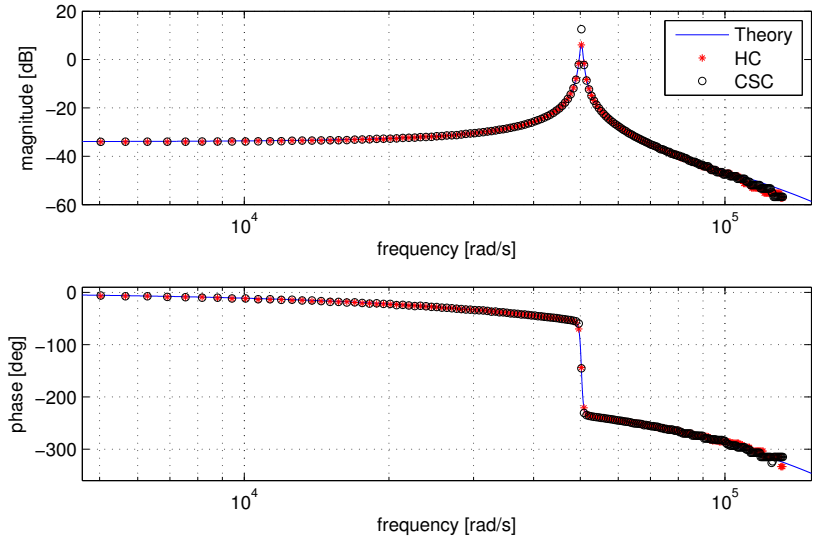

Figure 5.2: A comparison of heterodyne and conventionally sampled control of the cantilever emulator, with $Q = 1000$, $Qc = 50$ and $k = 50$

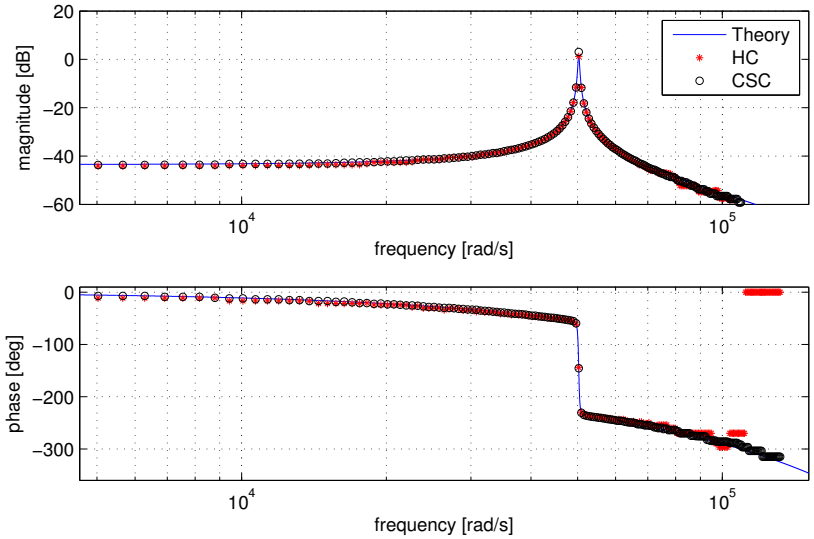

Figure 5.3: A comparison of heterodyne and conventionally sampled control of the cantilever emulator, with $Q = 1000$, $Qc = 150$ and $k = 150$

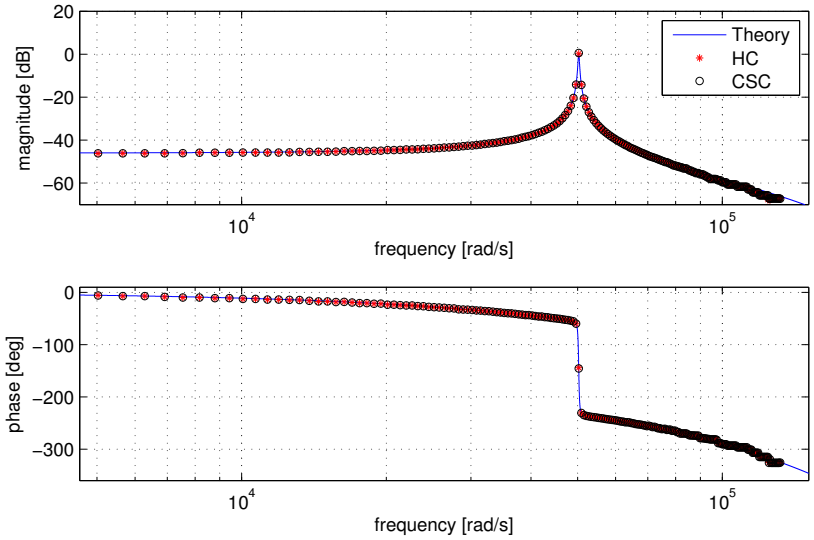

Figure 5.4: A comparison of heterodyne and conventionally sampled control of the cantilever emulator, with $Q = 1000$, $Qc = 200$ and $k = 200$

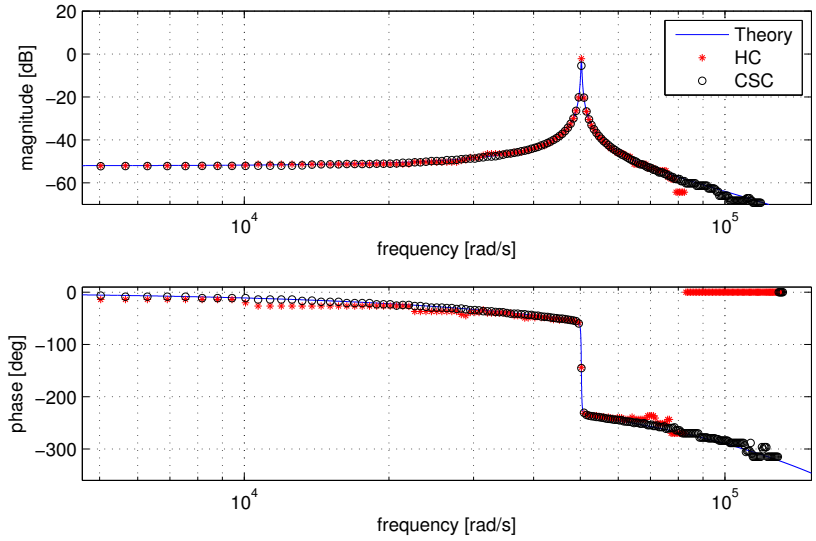

Figure 5.5: A comparison of heterodyne and conventionally sampled control of the cantilever emulator, with $Q = 1000$, $Qc = 400$ and $k = 400$

We see from these figures that the performance of the heterodyne and conventionally sampled control methods are nearly identical. Furthermore, the data from both methods match the theory nicely.

5.2 The Cantilever Control

From Section 5.1 it was determined that the SBP (using either the conventional or heterodyne control method) is capabale of controlling the cantilever emulator. Therefore, control of the physical microcantilever using the SBP should also be successful. This chapter presents data from testing of the SBP (using the heterodyne method) in control of the microcantlever and compares it to data gathered from control of the microcantilever with the USRP. This data was taken from the SR780 spectrum analyzer, which computed the power spectrum of the interferometer signal. Each of the figures below represent the spectrum with a different controlled quality. Additionally, the following parameters of the system were: $f_o = 7982.4$ Hz, $Q = 19342$, $K_i = -28.59$ mv/nm, $K_i K_{pa} K_a K_c = -7.17$ N/m, assumed mass = 6.98 ng, FPGA sample rate = 112045 Hz, and $n = 14$ samples per period.

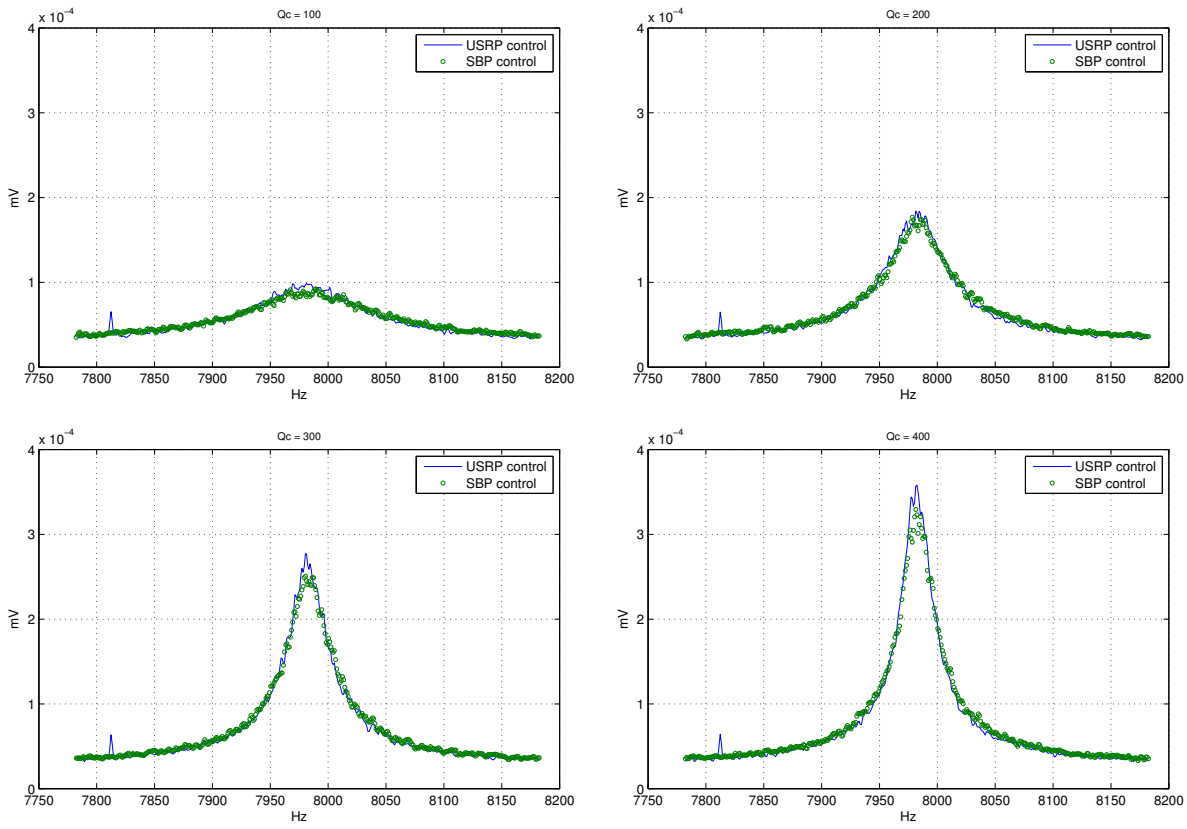

Figure 5.6: Comparison USRP controlled and SBP controlled microcantilever power spectrum from SR780. Controlled quality Q_c is varied from 100 to 400 in increments of 100

It can be seen form the data above, that the performance of the SBP in controlling the microcantilever is comparable to that of the USRP.

5.3 Spectral Analysis

Another important function of the SBP is spectral analysis of the microcantilever signal. This function is performed on the processor and uses the X and Y data collected from the FPGA. The mathematics needed to compute the power spectrum from the X and Y data is covered in Section 4.8. The following figures in this section present spectral data of the interferometer signal, calculated from two sources: the SR780 spectrum analyzer, and the SBP. In the first figure, the power spectrum is from the open-loop microcantilever. In the next four figures, the microcantilever is controlled with varying values of the controlled

quality.

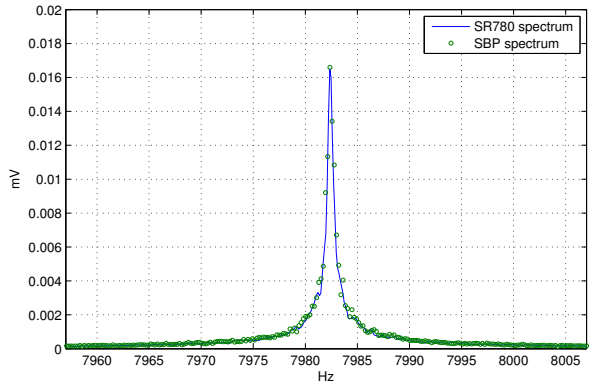

Figure 5.7: Open Loop Cantilever

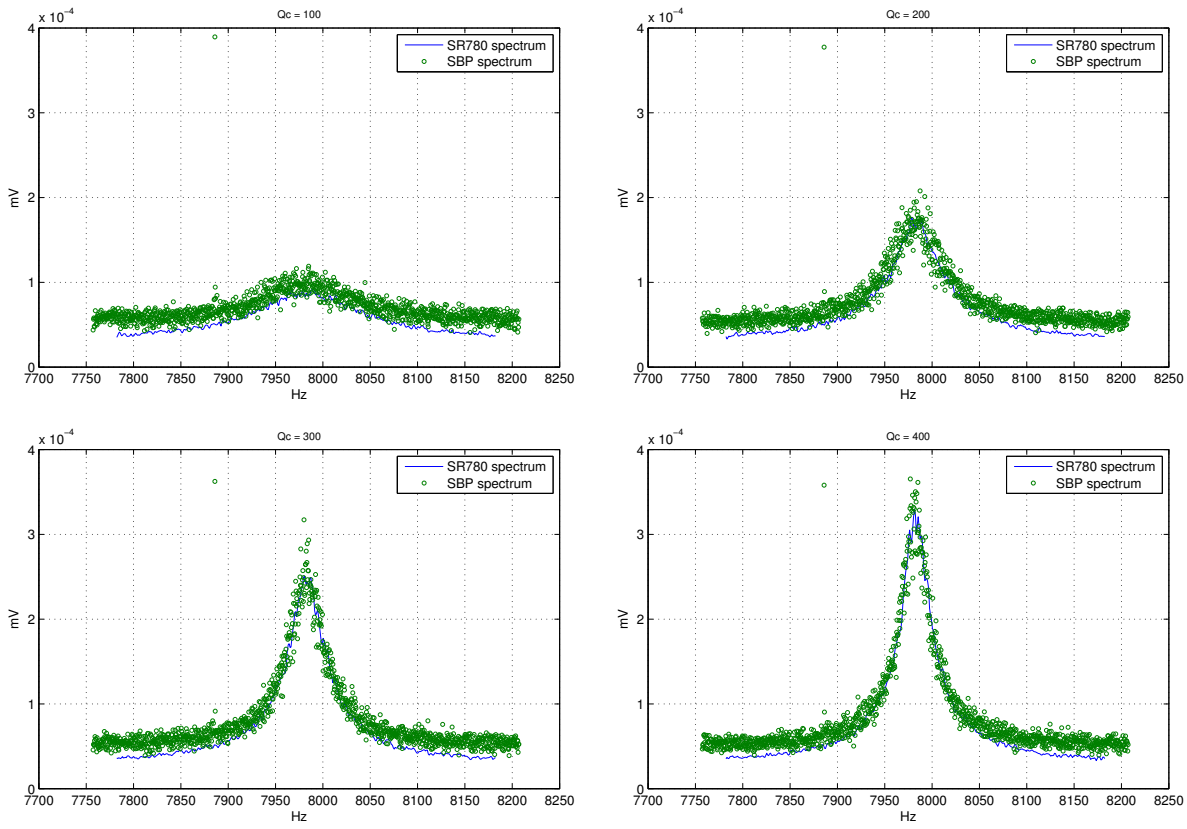

Figure 5.8: Comparison of SR780 and SBP spectral data of the SBP-controlled microcantilever. The controlled quality is varied from 100 to 400 in increments of 100

It can be seen from the data above, that the performance of the SBP in performing spectral analysis is comparable to that of the SR780.

5.4 Commanded Amplitude

A final key function of the SBP is amplitude command of the microcantilever as described in Section 4.9. This function is used to acquire phase-shift measurements in a MRFM experiment.

To test the effectiveness of this function while performing microcantilever control with the SBP, two parameters were measured: the amplitude of the microcantiever, and the error signal (as decribed in Equation 4.27). Figure 5.9 presents the commanded amplitude data, measured from two sources: a 8505A voltmeter (amplitude inferred through K_i), and the

SBP (amplitude inferred from X and Y data).

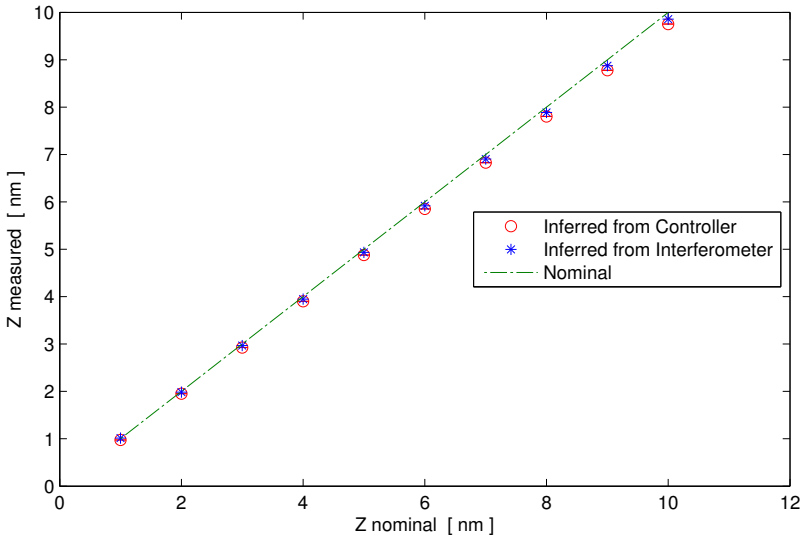

Figure 5.9: 8505A and processor inferred amplitudes compared with the nominal amplitude

Figure 5.10 presents the error data with error bars inferred from the X and Y data on the SBP. This data is compared with the theoretical error of Equation 4.27.

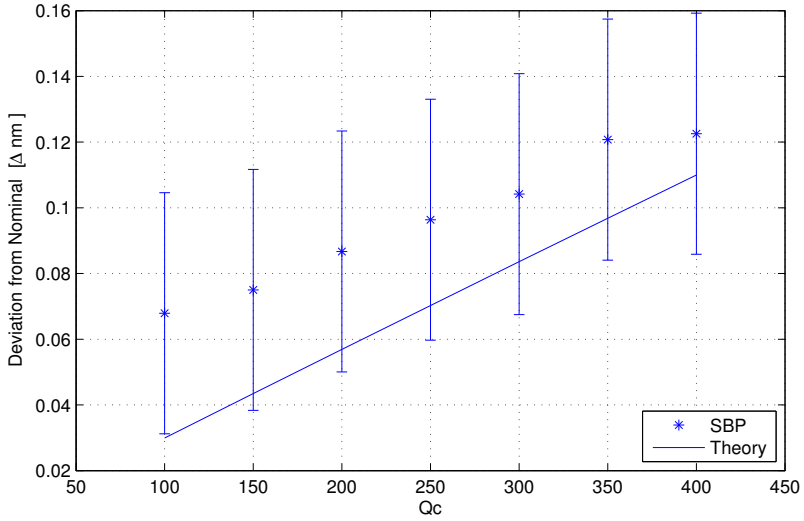

Figure 5.10: 8505A error signal compared with Equation 4.27

In the first figure, the results from both the 8505A and the processor are nearly identical. Additionally, this data match the nominal amplitude nicely. In the second figure, we see that the data and theory match nicely as well.

Chapter 6

CONCLUSION

In the original MRFM experiment at the University of Washington, functions such as microcantilever control, noise filtering, and signal generation were performed through an interconnection of single-function laboratory instruments. This configuration was not ideal, as it was difficult to diagnose and consequently expensive to build and maintain. Moreover, the operational programming for some of this equipment, such as the USRP, required considerable time and effort to design.

This project has explored the use of a single board platform (SBP), incorporating a real-time processor, and a FPGA, to perform many of the functions required for a MRFM experiment. A conventional computer hosts the SBP, with programming at all three levels (host, real-time processor, and FPGA) performed in the LabVIEW language: G. Furthermore, by implementing the SBP in the MRFM experiment, three key advantages were realized.

The first of these advantages was the consolidation of many functions onto the SBP. Some of these functions were: microcantilever control, spectral analysis, lockin amplification and synchronization signal generation. A second advantage was the reduction in complexity of the system. In previous MRFM experiments, the analog connections among the large number of discrete instruments often lead to increased noise and ground loops. By combining functions onto a single board, many of these analog connections have been eliminated. A final advantage has been an increase in programming efficiency. The task of developing experimental protocols has been greatly eased by using a common programming language at all levels, and across many functions.

Although the SBP has slower sampling rates than the earlier controller, the USRP, the experimental results presented in chapter 5 demonstrated the SBP was successful in performing the following tasks: (1) use as a cantilever emulator for testing control and signal

processing over a wide range of parameters, (2) microcantilever control using either the conventionally sampled or heterodyne control method, (3) spectral analysis of microcantilevers, and (4), amplitude command of the microcantilever. In all of these cases the performance of the SBP implementation met or exceeded that of the discreet-instrument approach.

This data also demonstrates the universality of SBPs in performing a wide range of computing tasks. This fact touches on a unique attribute of SBPs: recourses such as a real-time processor, FPGA, and analog and digital I/O reside on a single piece of hardware. As SBP technology advances, the range of tasks SBPs can perform will only improve, and perhaps someday their use in MRFM and many other research interests will become the norm.

Chapter 7

APPENDIX

7.1 Emulator Front Panel and Block Diagram

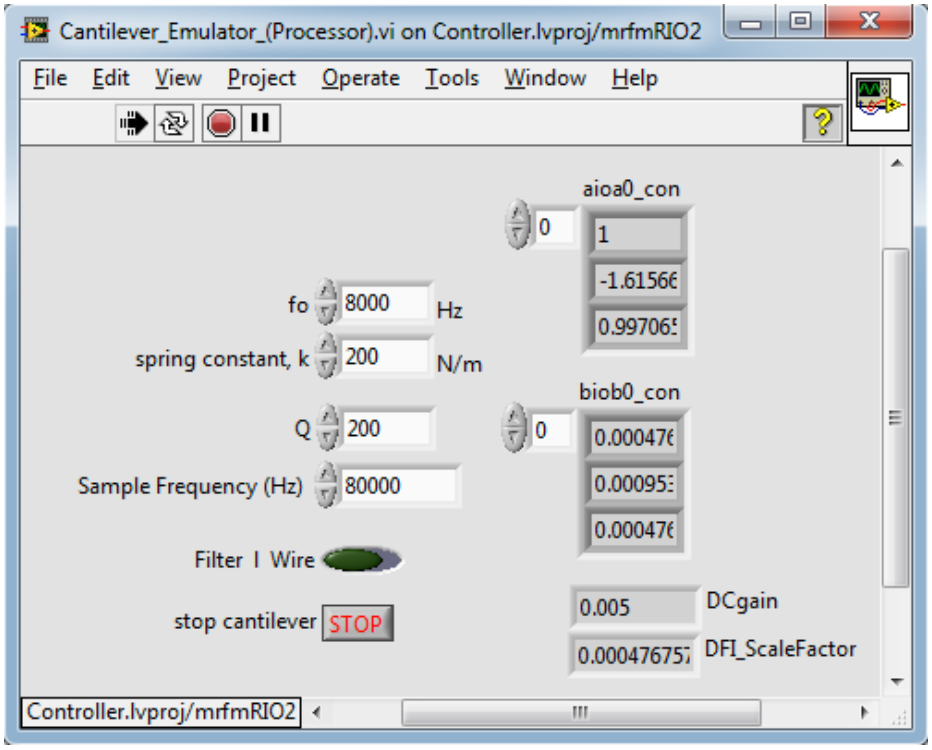

Figure 7.1: The LabVIEW Front Panel of the Cantilever Emulator

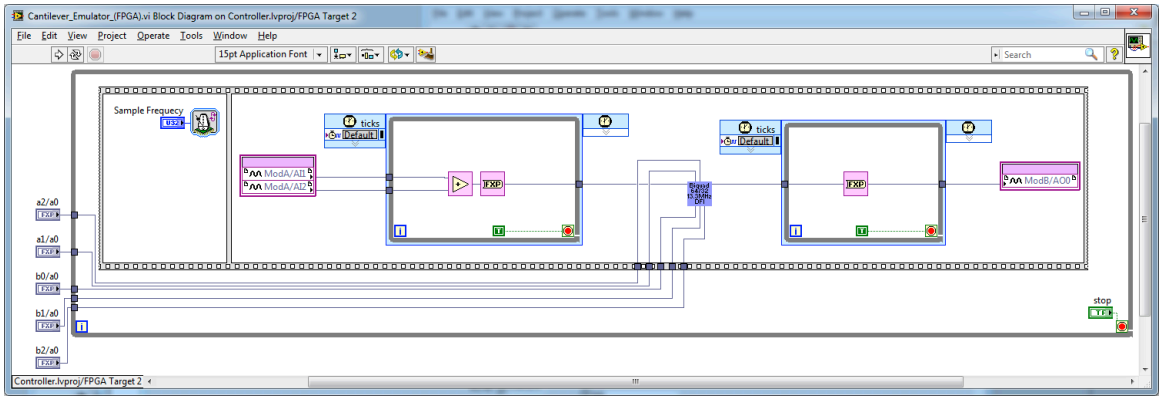

Figure 7.2: The LabVIEW Block Diagram of the Cantilever Emulator

7.2 Conventional Controller Front Panel and Block Diagram

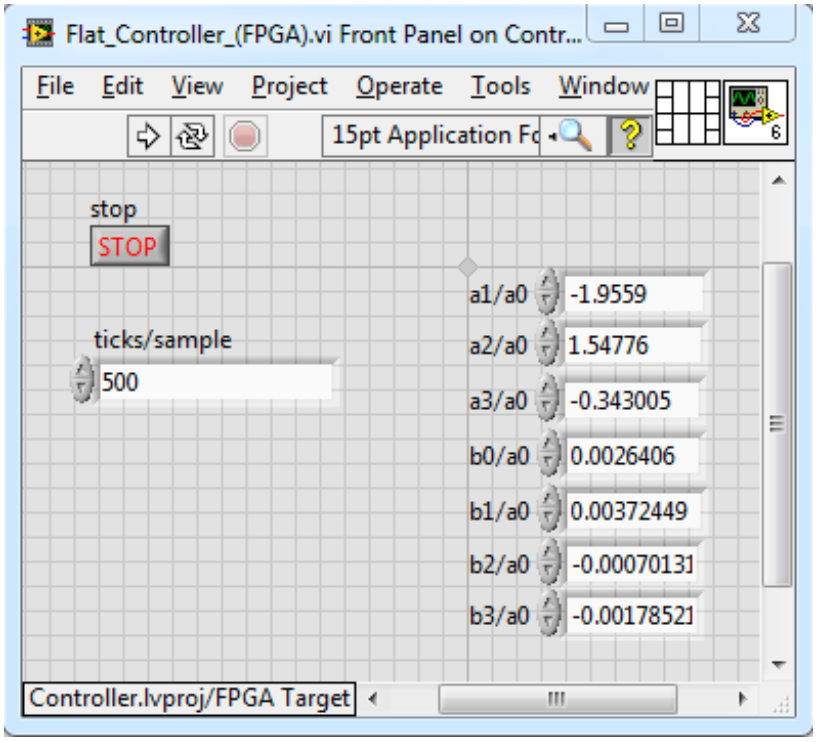

Figure 7.3: The LabVIEW Front Panel of the Phase-Lead Controller

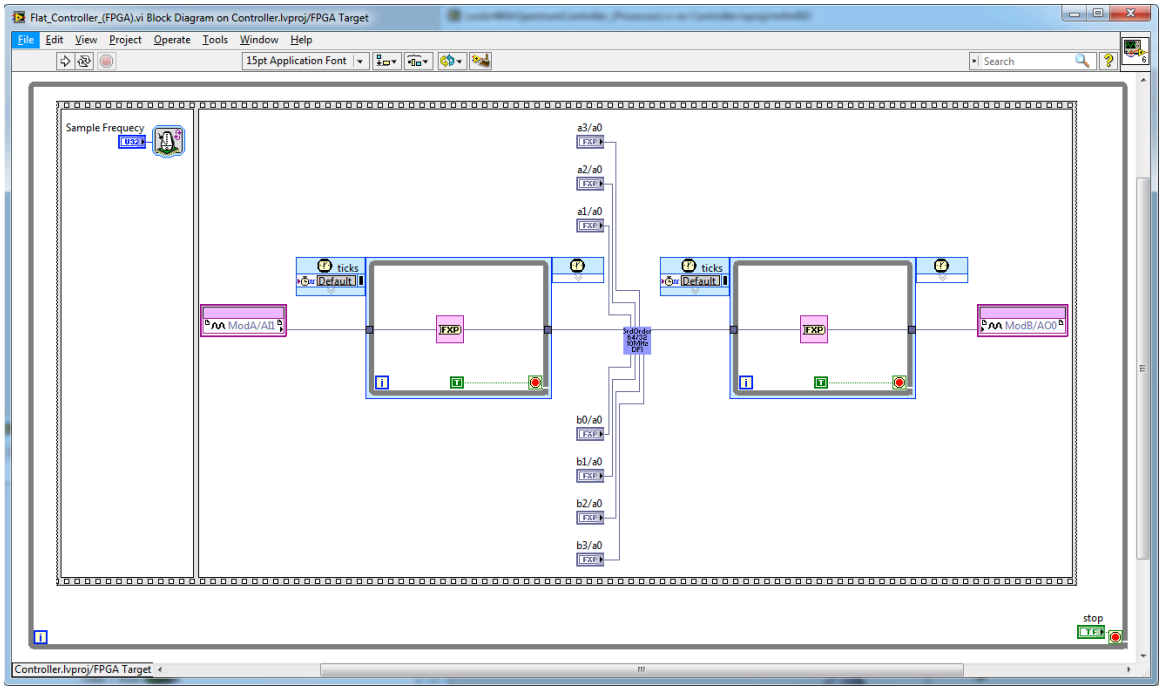

Figure 7.4: The LabVIEW Block Diagram of the Conventional Controller

7.3 Heterodyne Controller Front Panel and Processor Block Diagram

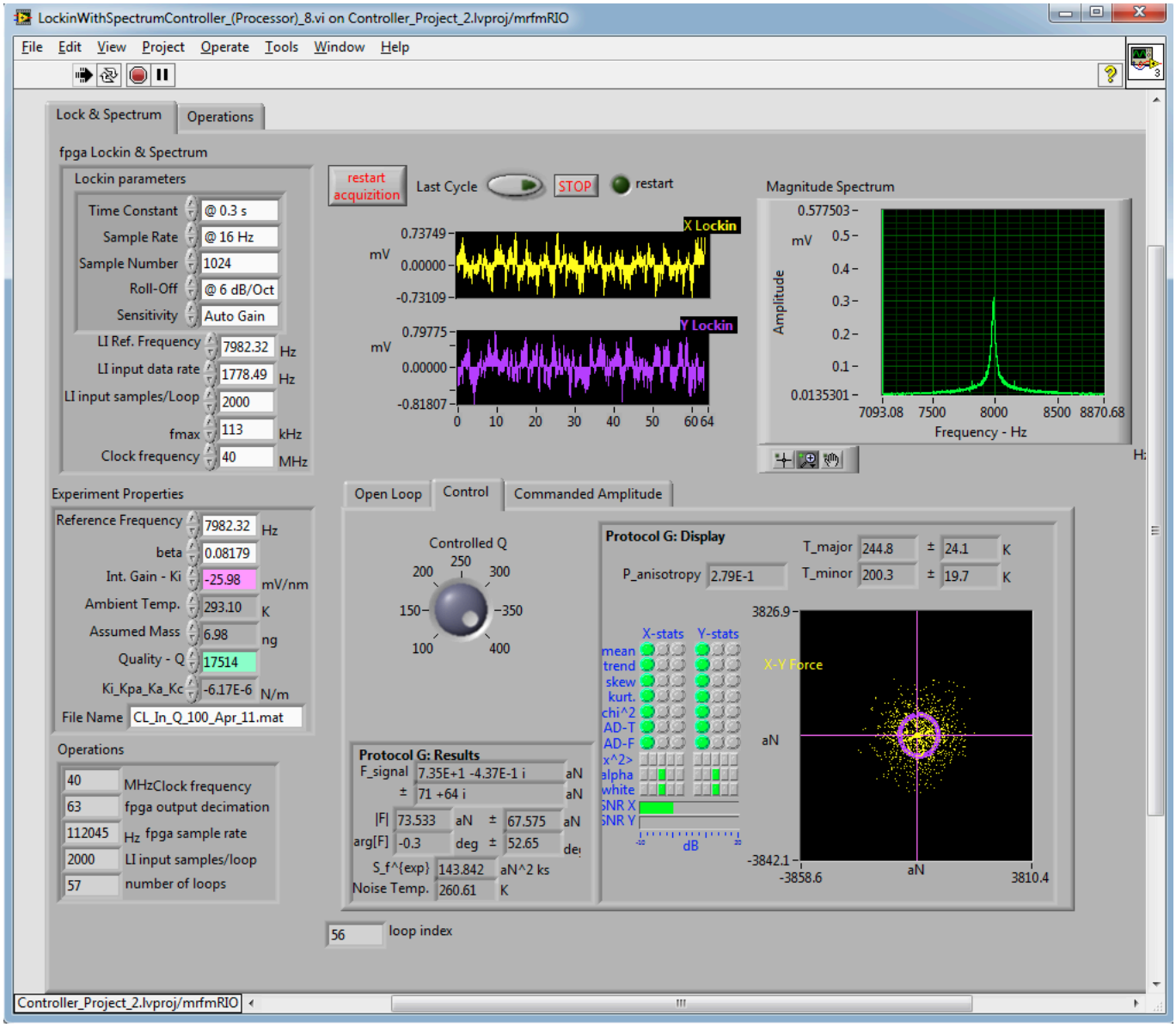

Figure 7.5: Heterodyne Controller Front Panel

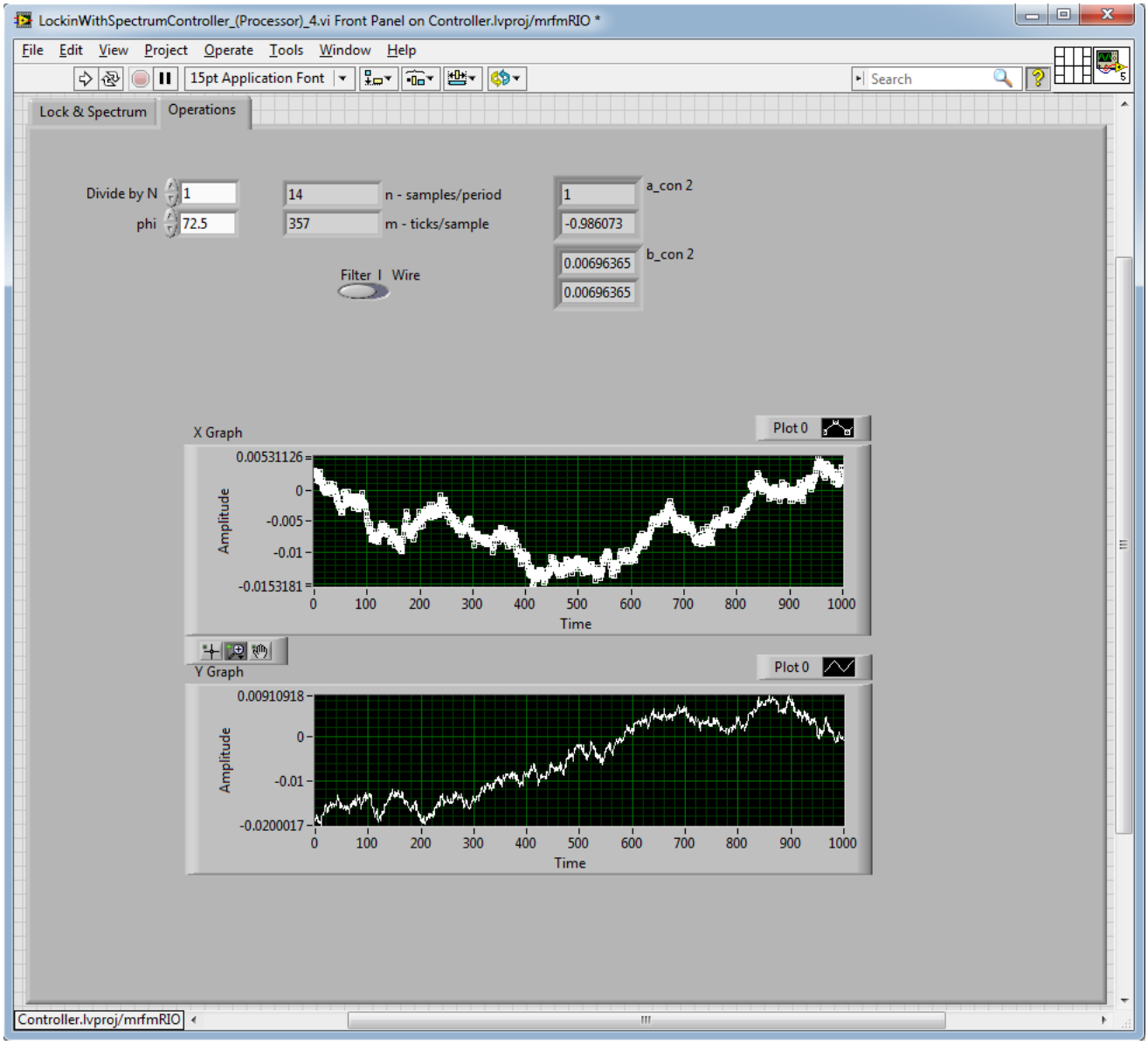

Figure 7.6: Heterodyne Controller Front Panel, Operations tab

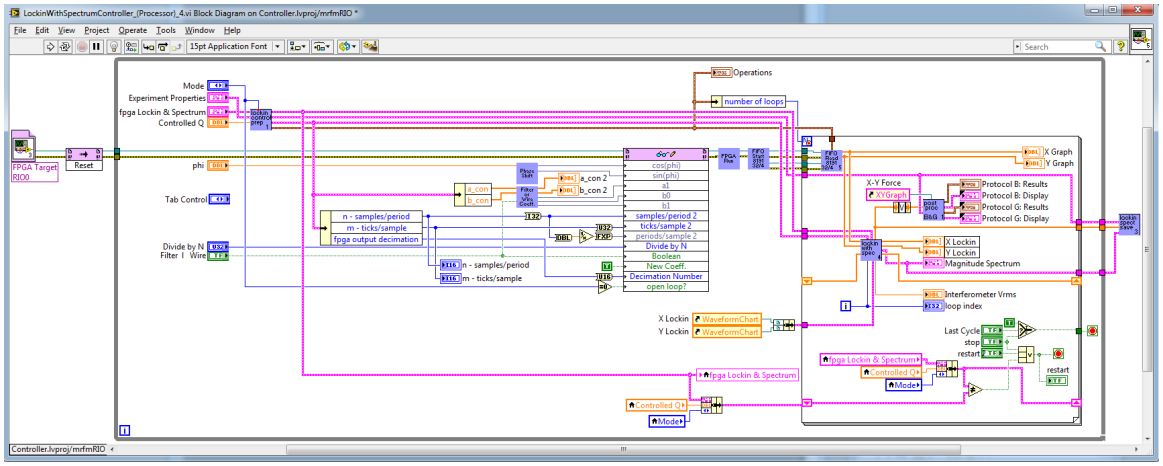

Figure 7.7: Heterodyne Controller, Processor Block Diagram

7.4 Testing Controller with MRFM Cantilever: Method

0. Run Protocols A - G

1. SFPD Controller

- Set Sampling Frequency: $f_s = 113kHz$ (this will produce a sampling frequency of about 112 kHz)

- Measure Phase Lead ϕ of Controller at 112 kHz:

  -Step 1. In "Operations" on SFPD Front Panel, set the boolean switch from "Filter" to "Wire"

  -Step 2. Connect the output of a lock-in amplifier to the input of SFPD controller, and connect the output of SFPD controller to the input of the lock-in

  -Step 3. Set the lock-in generated sine wave to the natural frequency of the MRFM cantilever (f_0), found in Protocol B. Note the phase shift on the screen $\phi_{Lock-in}$

-Step 4. In "Operation", set the boolean switch from "Wire" to "Filter", and set the phase lead:

$$\phi = \phi_{Lock-in} + \phi_{Int.} + \phi_{ConAmp}$$

$\phi_{Int.}$ and ϕ_{ConAmp} are found in Protocol D

- Set Natural Frequency f_0
- Set Calibrated Spring Constant:

$$k_{cal} = \frac{k}{k_i k_{pa} k_a k_c}$$

The denominator is found in Protocol D

k is found in Protocol B

- Set β, found in Protocol D
- Set Q found in Protocol A
- Set k_i found in Protocol B

2. Run Open-Loop Tests

Table 7.1: Open Loop

Protocols	SFPD
Acquire the Front Panel from Protocol B	Set lock-in parameters τ, N, roll-off and $f_{s,LI}$
Acquire the Spectrum from the SR780	Acquire V_x, V_y, and Spectral Analysis

Note the inferred temperature calculated by SFPD controller, found in the sub-VI "Lockin Spec Save".

3. Compare v_{rms} from SFPD and Protocol B

$$v_{rms} = x_{rms} \times k_i$$

4. Run Closed-Loop Tests (repeat for Q_c=100, 200 and 400)

Table 7.2: Closed Loop

Protocols	SFPD
Acquire the Front Panel from Protocol G	Set lock-in parameters τ, N, roll-off and $f_{s,LI}$
Acquire the Spectrum from the SR780	Acquire V_x, V_y, and Spectral Analysis

BIBLIOGRAPHY

[1] Thomas E. Kriewall II. *Heterodyne Digital Control and Frequency Estimation in Magnetic Resonance Force Microscopy.* PhD thesis, University of Washington, 2004.

[2] D. W. Clarke. On the design of adaptive notch filters. *International Journal of Adaptive Control and Signal Processing*, 15(7):715–744, 2001.

[3] J. L. Garbini, K. J. Bruland, W. M. Dougherty, and J. A. Sidles. Optimal control of force microscope cantilevers. i. controller design. *Journal of Applied Physics*, 80(4):1951–1958, 1996.

[4] "Heterodyne", Wikipedia.com, 2012. Web.

[5] Gene F. Franklin. *Digital Control of Dynamic Systems.* Pearson Education, Inc., third edition, 2002.

[6] Jonathan P. Jacky, Joseph L. Garbini, Matthew Ettus, and John A. Sidles. Digital control of force microscope cantilevers using a field programmable gate array. *Review of Scientific Instruments*, 79(12):123705, 2008.

[7] J.S. Bendat and A.G. Piersol. Random data: Analysis and measurement procedures. *Wiley-Interscience*, 1971.