Instrumentation and control of harmonic oscillators via a single-board microprocessor-FPGA device

Rico A. R. Picone, Solomon Davis, Cameron Devine, Joseph L. Garbini, and John A. Sidles

Citation: Review of Scientific Instruments **88**, 045108 (2017); doi: 10.1063/1.4979971
View online: http://dx.doi.org/10.1063/1.4979971
View Table of Contents: http://aip.scitation.org/toc/rsi/88/4
Published by the American Institute of Physics

Instrumentation and control of harmonic oscillators via a single-board microprocessor-FPGA device

Rico A. R. Picone,[1,a)] Solomon Davis,[2] Cameron Devine,[3] Joseph L. Garbini,[3,b)] and John A. Sidles[3,b)]
[1]*Department of Mechanical Engineering, Saint Martin's University, Lacey, Washington 98503, USA*
[2]*Faculty of Mechanical Engineering, Technion – Israel Institute of Technology, Haifa, Israel*
[3]*Department of Mechanical Engineering, University of Washington, Seattle, Washington 98195, USA*

(Received 23 December 2016; accepted 23 March 2017; published online 17 April 2017)

We report the development of an instrumentation and control system instantiated on a microprocessor-field programmable gate array (FPGA) device for a harmonic oscillator comprising a portion of a magnetic resonance force microscope. The specific advantages of the system are that it minimizes computation, increases maintainability, and reduces the technical barrier required to enter the experimental field of magnetic resonance force microscopy. Heterodyne digital control and measurement yields computational advantages. A single microprocessor-FPGA device improves system maintainability by using a single programming language. The system presented requires significantly less technical expertise to instantiate than the instrumentation of previous systems, yet integrity of performance is retained and demonstrated with experimental data. *Published by AIP Publishing.* [http://dx.doi.org/10.1063/1.4979971]

I. INTRODUCTION

It has long been observed that the harmonic oscillator appears ubiquitously in physical models.[1] More recently, the harmonic oscillator has become the centerpiece of a number of measurement technologies, notably in detectors of the gravitational constant,[2–4] magnetic resonance imaging, and scanning probe microscopy.[5–7] Due to their dynamic response similarities, the instrumentation and control techniques for these detectors can be shared among these otherwise disparate research communities. We report an instrumentation and control system for a damped harmonic oscillator that is the detector for magnetic resonance force microscopy (MRFM), although the system will find wider applications.

MRFM is a scanned-probe method of mechanically detecting extremely small magnetic resonance signals and reconstructing corresponding images.[7–9] Three-dimensional image resolutions of 4 nm have been achieved,[10,11] and recent experimental and theoretical work on sample hyperpolarization may lead to even greater resolution.[12–14] The detection of the statistical polarization of proton spins, which derives from random fluctuations, has been studied to good effect,[10,11,15] but a recent signal-to-noise analysis[12] suggests that there are distinct advantages to hyperpolarized detection.

MRFM experiments require a number of instruments (typically), including a digital controller, a lock-in amplifier, a spectrum analyzer, and a signal generator. Digital control systems typically used require deep digital hardware understanding and are difficult to maintain. Furthermore, the other instrumentation listed above creates a barrier to entry to the field of MRFM. It is the purpose of this report to describe the authors' maintainable, relatively easy-to-implement (low-cost) system instantiated in a microprocessor-field programmable gate array (FPGA) device that functions in all four roles: digital controller, lock-in amplifier, spectrum analyzer, and signal generator. Details of the apparatus—hardware and software—are described in Section IV.

Due to the advantageous noise scaling of mechanical oscillators, ultrasensitive microcantilevers detect MRFM signals. The high mechanical quality Q and low stiffness k of these microcantilevers are indicated due to the sensitivity-limiting Brownian motion of the microcantilever, which has spectral density inversely proportional to Q and directly proportional to $\sqrt{k}$.[16] However, the higher the microcantilever quality, the narrower the detection bandwidth, which can be problematic for detection. Furthermore, the lower the microcantilever stiffness, the larger the mean-square Brownian amplitude of oscillation. This motion can be detrimental to MRFM experiments in two manners: (1) if the detector oscillates at higher amplitude, the image resolution is lost, and (2) the microcantilever may be unable to withstand the stresses associated with the large amplitudes of oscillation. For these reasons, feedback control is used on the motion of the microcantilever to broaden the detection bandwidth and reduce the amplitude of oscillation.

MRFM experiments have been performed with field programmable gate array (FPGA) based microcantilever controller hardware,[17] but this hardware was, until recently, rather non-standard and required a great deal of expertise to implement and maintain. The recent introduction of FPGA-based hardware (such as National Instruments' Single-Board "RIO" devices), which include both a microprocessor and an FPGA, allows the application and maintenance of FPGA-based control without a deep knowledge of digital hardware. In fact, a single programming language can be used from top-to-bottom,

a)Also at Department of Mechanical Engineering, University of Washington, Seattle, Washington 98195, USA. Electronic mail: rpicone@stmartin.edu.
b)Also at Institute for Soldier Healing, Seattle, Washington 98105, USA.

including the programming of the FPGA. Despite these advantages, there remain several important considerations when implementing a controller in the microprocessor-FPGA devices. Some of these are described in Section IV.

Heterodyne digital control is a control technique that can reduce control computation and has been applied to MRFM with success.[18] It will be shown in Section IV that this technique reduces computation for instrumentation functions as well. Heterodyne control is introduced in Section II, which is followed by a description of the apparatus—hardware and software—in Section IV, and the system performance is evaluated in Section V.

II. HETERODYNE DIGITAL CONTROL

Heterodyne control for MRFM was originally developed to control exceptionally high resonance frequency (1–10 MHz) microcantilevers[19] with relatively low computational loads. Although high-frequency microcantilevers have not yet been developed for MRFM, the reduction of computational load is a significant advantage for instantiating control of conventional MRFM microcantilevers—with typical resonance frequencies around 10 kHz—in microprocessor-FPGA boards. The computational advantage is primarily due to the fact that the heterodyne control computation includes processing the signal in such a way that several instrumentation functions can leverage the same processed signal. In this way, microcantilever control and several MRFM instrumentation functions can be performed by a single device.

The key signal processing computation that can be leveraged for instrumentation and control is *heterodyning*, a technique in which two signals are mixed (multiplied) and the resulting signal has new frequency components at the sum and difference of the input signal frequencies. In MRFM, the microcantilever's position signal is dominated by frequency components very near to its resonance frequency $\omega_r \in \mathbb{R}$. Multiplying this signal by a sinusoid at (or near) ω_r results in a signal that has two spectral copies of the original signal—one at $2\omega_r$ and one at DC. Applying a low-pass filter eliminates the doubled copy and leaves the DC copy (note that there is an amplitude scaling factor to consider). This is called the *in-phase* discrete signal $x : \{t_0, t_0 + \delta t, t_0 + 2\delta t, \ldots\} \to \mathbb{R}$, where $t_0 \in \mathbb{R}$ is an initial time and $\delta t \in \mathbb{R}$ is the time between samples. Similar to and in parallel with this process, mixing the original signal with a sinusoid 90° out of phase with the in-phase sinusoid and low-pass filtering, the result produces the *quadrature* signal $y : \{t_0, t_0 + \delta t, t_0 + 2\delta t, \ldots\} \to \mathbb{R}$. Together, the creation of in-phase and quadrature signals is called *downconverting* or *downmixing*.

The downmixed signals together are essentially a very clean copy of a band of the original signal's spectrum centered about the microcantilever's natural frequency, now centered about DC. It is precisely this function that a lock-in amplifier performs, which is why most MRFM signals are processed with a lock-in amplifier. It is this downmixed signal that can be used to compute the control signal and several measurements of interest in an MRFM experiment, each of which is typically the function of separate instruments. In this section, we consider only the control scheme, leaving for Secs. III and IV the instrumentation and hardware functionality.

The control scheme adopted here and shown in Fig. 1(a) takes advantage of the fact that the optimal controller transfer functions (one for the in-phase signal and another for the quadrature signal) for the system also act as low-pass filters. This allows us to apply control while *downmixing*, minimizing computation by avoiding separate filter and control applications.

The controller is a linear quadratic regulator that is designed to produce the desired microcantilever dynamics. The method of applying control varies by MRFM apparatus, with some moving the base of the microcantilever via piezoelectric actuation[10,12] and others applying a magnetic field to a magnetized particle affixed to the tip of the cantilever via a nearby coil. The latter method is employed here.[20] Early MRFM experiments required the microcantilever motion to be simply regulated,[21] but recent MRFM and non-contact atomic force microscopy (nc-AFM) experiments have included a

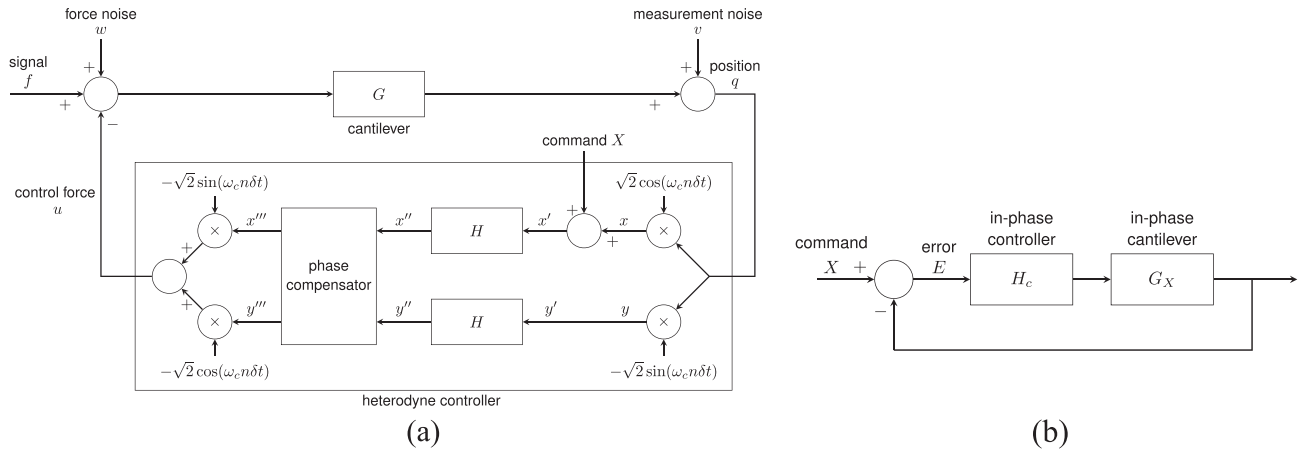

FIG. 1. Control block diagrams. (a) A block diagram of the cantilever feedback control loop with heterodyne control. The MRFM signal is detected by the microcantilever *G*, which is measured to have the position q. The feedback loop includes the heterodyne controller, which downmixes the signal, applies optimal control with an amplitude command and phase compensation, and upmixes the signal before being fed back via the control coil of Fig. 2. (b) A simplified block diagram highlighting feedback control for error analysis.

driven oscillation at a specific amplitude.[8,22] In the present scheme, driving the microcantilever to a specific amplitude is easily achieved by adding the desired amplitude X to the in-phase signal before applying the control filter, as shown in Fig. 1(a). The signals that result are $x' = x + X$ and $y' = y$.

Let $\alpha \in \mathbb{R}$ be a parameter set by the maximum allowable microcantilever displacement and maximum available control force and $\beta \in \mathbb{R}$ be a parameter that represents the quality of the state estimation (determined by the relative quantities of process and measurement noise).[23] Furthermore, let $Q \in \mathbb{R}^+$, $k \in \mathbb{R}^+$, and $\omega_n \in \mathbb{R}^+$ be the microcantilever quality, spring constant, and natural frequency. Finally, let a low-pass filter gain K_H and cutoff frequency ω_H be defined as

$$K_H = k\alpha\beta\omega_n/2 \quad \text{and} \tag{1a}$$

$$\omega_H = \omega_n \left(1 + Q(\alpha + \beta)\right)/Q. \tag{1b}$$

A continuous-time optimal controller that will be discretized and applied to each signal x' and y' is given by the transfer function $H_c : \mathbb{C} \to \mathbb{C}$, defined by[18,24]

$$H_c(s) = \frac{K_H}{s + \omega_H}. \tag{2}$$

To apply Tustin's method with prewarping[25] to match the frequency response at the natural frequency ω_n of the microcantilever, the substitution into (2) is

$$s \leftarrow \frac{\omega_n}{\tan(\omega_n \delta t/2)} \cdot \frac{z-1}{z+1}. \tag{3}$$

Let x'' and y'' be the controller output signals of the x' and y' channels, respectively. The difference equation corresponding to the discrete transfer function H found through substitution (3) for the in-phase signal (for the quadrature signal, it is identical with $x' \leftarrow y'$ and $x'' \leftarrow y''$) is

$$x''(n) = -\sum_{k=1}^{3} \frac{a_k}{a_0} x''(n-k) + \sum_{k=0}^{3} \frac{b_k}{a_0} x'(n-k), \tag{4}$$

where the coefficients $a_k \in \mathbb{R}$ and $b_k \in \mathbb{R}$ are found from (2) with the substitution (3).[26] This (4) can be used for digital implementation of the controller.

A. Phase compensation

The position of the microcantilever is measured by an interferometer and the optical signal is converted to an electrical signal via a photodiode. The photodiode introduces a significant amount of phase offset, and additional phase is contributed by the A/D and D/A conversions. Instead of raising the order of the filters by adding phase-lead compensation to the controller, the phase can be manipulated directly in the mixed-down signal. Let x''' and y''' be the output signals of the phase compensator and $\phi \in \mathbb{R}$ be the desired amount of phase compensation. The phase compensator computes its outputs via the simple formulae

$$x''' = x'' \cos\phi - y'' \sin\phi \quad \text{and} \tag{5a}$$

$$y''' = x'' \sin\phi + y'' \cos\phi. \tag{5b}$$

The final step is to convert the signals x''' and y''' to the control signal u. This process is called *upconverting* or *upmixing* and is computed from the expression

$$u = -\sqrt{2}x''' \sin(\omega_r n\delta t) - \sqrt{2}y''' \cos(\omega_r n\delta t). \tag{6}$$

The software and hardware instantiation of the heterodyne control scheme is described in Sec. IV.

B. Amplitude error analysis

The difference between the commanded amplitude and the actual amplitude can be theoretically predicted by the following analysis. With the assumption that the in-phase signal dominates, which is the case when, as is typical, the commanded amplitude is much larger than both the noise (w and v) and the signal (f), the simplified block diagram of Fig. 1(b) shows the effective feedback loop for the amplitude control. Kriewall *et al.*[18] showed that the microcantilever dynamics can be considered closed-loop in the downconverted in-phase and quadrature domain. Let $G_X(s)$ be the continuous-time transfer function representing the in-phase dynamics, then[27]

$$G_X(s) = \frac{Q/k}{(2Q/\omega_n)s + 1}, \tag{7}$$

where $k \in \mathbb{R}^+$ is the microcantilever spring constant, Q is its uncontrolled quality, and ω_n is its natural frequency. This in-phase microcantilever model enables an "all downmixed" perspective of position control, shown in Fig. 1(b).

A block-diagram analysis yields the continuous transfer function $T(s)$ from the command to the error,

$$T(s) \equiv \frac{E(s)}{X(s)} = \frac{1}{1 + G_X(s)H_c(s)}. \tag{8}$$

Inserting (2) and (7) into (8) and taking the limit as s approaches zero, we obtain an expression for the DC-component of the error transfer function,

$$T(0) = \frac{1 + Q(\alpha + \beta)}{(1 + Q\alpha)(1 + Q\beta)}. \tag{9}$$

Since the mixed-down command amplitude X is completely DC—that is, $X = X(0)$—the inherent error for a commanded amplitude X is predicted to be

$$E(0) = T(0)X, \tag{10}$$

where $T(0)$ is given by (9). See Sec. V and Fig. 6(c) for a comparison of this prediction with experimental results.

III. HETERODYNE INSTRUMENTATION

The Single-Board RIO performs three key instrumentation functions, two of which are signal processing and the third of which is sine-wave generation for the downconversion and upconversion processes.

A. Lock-in amplification

Heterodyne downconversion is precisely the function of a lock-in amplifier, which is used to "lock-in" to signals at or near a known reference frequency and reject noise. MRFM signals, which are all near the microcantilever's resonance frequency ω_r are typically processed with lock-in amplifiers. It is therefore convenient that the heterodyne control computes x'' and y'', which—with a correction for the command amplitude—are essentially identical to the signals a lock-in amplifier would output.[28]

The FPGA computes these signals as part of the control, and it periodically passes downsampled versions to the processor. Details of the implementation are described further in Sec. IV.

B. Narrow-band heterodyne spectral analysis

The processor receives downsampled data from the FPGA and performs a fast Fourier transform (FFT) on each channel to generate the spectral estimate. If an FFT were to be performed on the unmixed data, a much larger bandwidth would be required and therefore at a much higher sample rate. However, the mixed-down data can be sampled at a much lower rate due to most of the signal being near DC.

Consider a continuous signal $c(t)$ constructed from an in-phase signal $a(t)$ and quadrature signal $b(t)$ such that

$$c(t) = a(t)\cos\omega_r t + b(t)\sin\omega_r t. \tag{11}$$

From the modulation property of the Fourier transform, the transform of c, $C(\omega)$, can be expressed in terms of the transforms of a and b, $A(\omega)$ and $B(\omega)$, respectively, as

$$C(\omega) = \frac{1}{2}A(\omega+\omega_r) + \frac{1}{2}A(\omega-\omega_r) + \\ -\frac{j}{2}B(\omega+\omega_r) + \frac{j}{2}B(\omega-\omega_r), \tag{12}$$

where $j = \sqrt{-1}$. Estimates of A and B can be combined via (12) to yield an estimate of C. Analogously, using the FFT algorithm on the in-phase and quadrature signals x''' and y''' to yield discrete complex signals X and Y, we can apply this result directly to estimate the one-sided power spectral density of the corresponding time-domain signal u after N samples spaced δt element-wise with the expression

$$\tilde{G}_k = \frac{2\delta t}{N}|U_k|^2, \tag{13}$$

where $k \in (0, 1, \ldots, N/2)$ and where the estimated transform U_k of u is

$$U_k = \frac{1}{2}X_k - \frac{j}{2}Y_k, \tag{14}$$

where the spectral center frequency is understood to be shifted in an angular frequency from DC to ω_r.

For a typical MRFM experiment, approximately 50 times fewer samples are required to compute the spectrum with this method. The FFT algorithm has complexity $\mathcal{O}(N \log N)$, so the total computational savings of this spectral analysis is a factor of about 70.

C. Sine-wave generation

In order to perform the heterodyne computation, two sinusoids must be computed: one synchronous with an external reference signal at the microcantilever resonance frequency and one ninety degrees out-of-phase. For efficient computation, a look-up table of sinusoidal values equally spaced in time over one period is externally triggered each reference period.

The look-up table can be computed once for a given reference frequency $f_r = \omega_r/2\pi$. Care must be taken when computing the number of FPGA clock cycles per sample $M \in \mathbb{Z}^+$ and the number of samples per reference period $N \in \mathbb{Z}^+$. Let f_c be the clock frequency and f_s be the nominal sample rate (typically as high as the hardware allows). The choices of M and N set the effective sample rate $\tilde{f}_s$ and the generated sinusoidal frequency $\tilde{f}_r$,

$$\tilde{f}_s = f_c/M \quad \text{and} \tag{15a}$$

$$\tilde{f}_r = f_c/MN, \tag{15b}$$

and it is desirable that $\tilde{f}_s \approx f_s$ and $\tilde{f}_r \approx f_r$. It can be shown that choosing M and N such that $|f_s - \tilde{f}_s|$ is minimized—as seems reasonable—can lead to signals with significant spectral components at higher harmonics.[29] The following values reduce these higher harmonics:

$$N = \text{floor}\, f_s/f_r \quad \text{and} \tag{16a}$$

$$M = \text{floor}\, f_c/Nf_r. \tag{16b}$$

An FPGA with a greater capacity (such as are available on different models) would be able to store a lookup table with multiple sinusoid periods. This would further reduce the introduction of higher harmonics.

IV. APPARATUS

The experimental apparatus varies for each MRFM apparatus, each of which is custom-built. Fig. 2 shows a schematic of a typical MRFM apparatus. A cryostat cools the sample, microcantilever, control and RF coils, and other mechanical components to temperatures that range from millikelvin[30] to 10 K, with 4-10 K being typical. Sample positioning, RF signal generation and modulation, cantilever position control, and signal processing instrumentation typically occur on the laboratory bench.

In the current instantiation, much of the instrumentation and digital control is performed on a Single-Board RIO sbRIO-9632 from National Instruments. However, separate sample positioning, RF generation and modulation, and user interface hardware are still required.

A. Software architecture

An advantage to this type of instrumentation and control is that a single programming language may be used to develop the software, including that which is deployed to the FPGA. Although the FPGA code requires a certain degree of expertise to write, it is significantly less than that required to program most stand-alone FPGAs.

Two simultaneous programs run on the board: one on the FPGA, primarily concerned with heterodyning and control, and the other on the processor, primarily concerned with instrument functions and data processing. We introduce the FGPA and processor programs in Secs. IV A 1 and IV A 2; supplementary detailed descriptions are included in Appendix A. The code is available as an open-source project (see Appendix B).

1. FPGA software architecture

The FPGA software required careful design to avoid overflowing the 46 080 logic cells of the FPGA. The two primary functions of the FPGA are to perform heterodyning and to

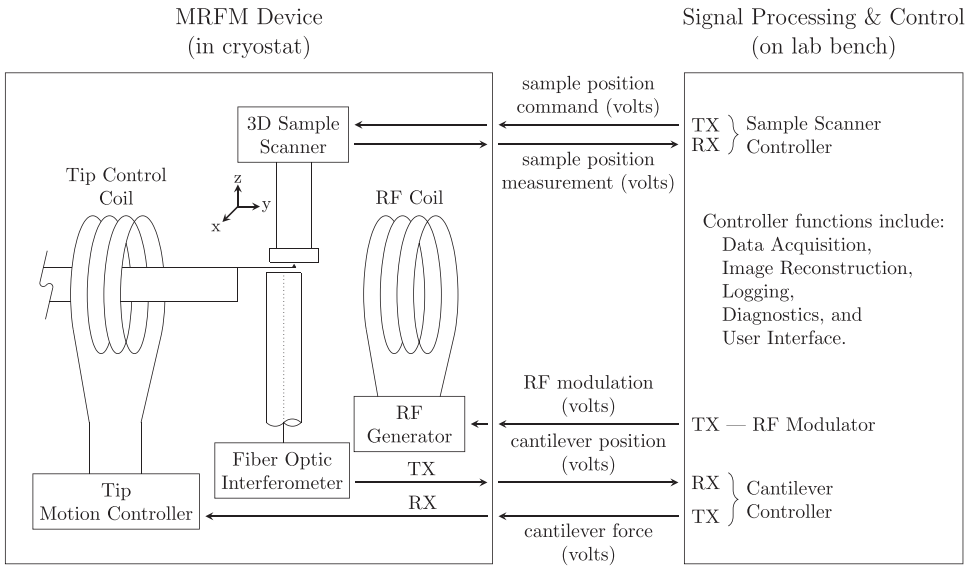

FIG. 2. A schematic of a typical MRFM apparatus. The cryostat contains a sample, a sample scanner, a magnet-tipped cantilever beam, a control coil, and a RF coil. On the laboratory bench, there is instrumentation that performs signal generation and processing. In the present system, several of the latter functions are performed via a single-board microprocessor-FPGA device.

apply control. Fig. 3 shows the architecture at a high level. Starting at block `f1` and proceeding with the arrow, in block `f2`, a loop iterator `j` is initialized. Decision block `f3` tests to see if the digital input `DIO0` is low or high. When an external function generator TTL signal connected to DIO0 is high, a sequence of events is triggered in the FPGA to generate synchronous sinusoids.

For some MRFM experiments, radiofrequency (rf) signals are applied that are synchronous with the microcantilever resonance frequency. In some cases, undesired heating can result; at times this can be mitigated by pulsing the rf signal once every `N` cantilever periods. This is the function of the `f4` output block: `DIO1` is used to trigger an rf pulse once every `N` microcantilever periods.

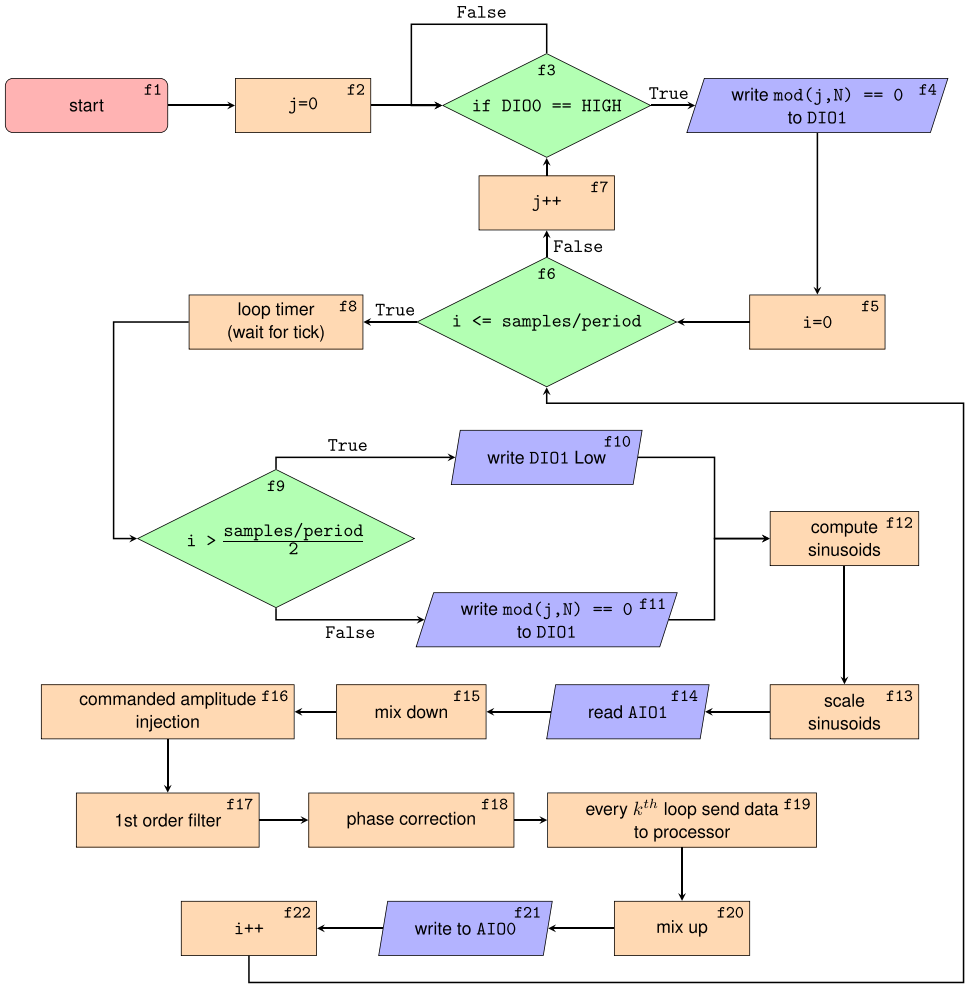

FIG. 3. FPGA software architecture flowchart.

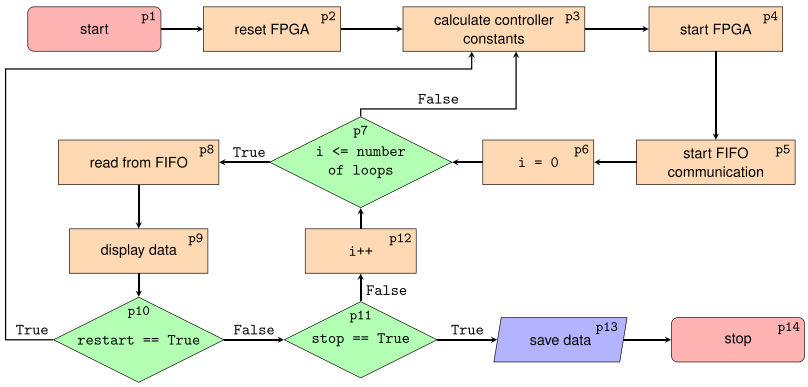

FIG. 4. Microprocessor software architecture flowchart.

The lower loop iterator is initialized in `f5`, and while the decision in `f6` as to whether or not `i` is less than or equal to the number of samples per period is true, the lower loop executes. Loop timing is critical, so a specialized loop timing function is called in block `f8`. This function knows how frequently the loop is supposed to execute and waits for the next "start time," unless that has already passed, in which case it begins the iteration immediately and adjusts the next "start time" accordingly.

Blocks `f9`, `f10`, and `f11` function such that the rf signal pulse trigger `DIO1` is high during the first half of a microcantilever period if and only if the current period is the `N`th.

Heterodyning requires the computation of a sine and a cosine function for each sample, which occurs in block `f12`. It is critical that the functions are computed with the correct frequency, yet a look-up table is required for performance; in this regard, Sec. III C describes some key considerations. Finally, the sine and cosine signals are scaled by $\sqrt{2}$ in block `f13`. Other important details for implementation, including fixed-point arithmetic considerations, are discussed in Appendix A.

The cantilever position measurement signal q is read from the analog input `AIO1` in block `f14`. The downconversion process is begun in `f15` by multiplying the measurement signal by each of the sine and cosine waves, generating signals x and y, respectively.

In accordance with the discussion in Sec. II, block `f16` adds the commanded microcantilever oscillation amplitude to the in-phase channel, before applying the controller transfer functions to each channel by passing them each through the difference Equation (4) in block `f17`. Finally, the control application is complete with the phase adjustment in `f18`.

The high sample rate of the FPGA is important for control, but is unnecessary and overwhelming for the processor. However, the instrumentation functions of the processor do not require such a high sample rate, and therefore, the FPGA decimates the data by selectively sending occasional samples to the processor.

The upconversion process of block `f20` re-uses the generated sinusoids of `f12`, and the result is applied via the analog output `AIO0` in block `f21`. All that remains is to iterate, as represented by `f22` and the return of the flowchart to `f6` to determine if the iteration remains within the range of a single microcantilever period.

2. Processor software architecture

The microprocessor-executed software performs higher-level functions than the FPGA-executed software. With reference to Fig. 4, the program begins by resetting the FPGA (`p2`) and computing the controller constants (`p3`) before starting the FPGA (`p4`) and FIFO communication (`p5`). The main for loop (`p7`) reads data from the FPGA (`p8`) and displays it (`p9`) on the user interface front-panel. The high sample rates required for control are not required for signal acquisition, so the processor receives only a downsampled signal from the FPGA. Upon completion of the data acquisition, the data are saved in block `p13`.

V. PERFORMANCE

Previously, the apparatus had used a universal software radio peripheral (USRP) to control the cantilever motion.[17] The USRP controller requires extensive digital electronics expertise to implement and maintain; moreover, it requires several external instruments in order to perform its functionality. In this section, the performance of the microprocessor-FPGA device is compared to the USRP controller performance.

Before considering controller performance, the performance of the microprocessor-FPGA spectral estimate for narrow-band processes is evaluated by comparing it to that of a Stanford Research Systems SR780 spectrum analyzer. For this purpose, open-loop cantilever interferometer signal power spectra are displayed in Fig. 5(a). The spectra are very similar for this narrow-band process.

Performance of the controller is first investigated by comparing open-loop microcantilever interferometer signals with the USRP controller and with the microprocessor-FPGA controller, which were measured with the SR780 spectrum analyzer and are plotted in Fig. 5(b). The controller implemented in each device was for a resonance frequency of 7982.4 Hz, and at this resonance frequency, the plot shows that the performances of the devices were nearly identical.

The performance of the controller to oscillate the microcantilever at a commanded amplitude of oscillation is described in Fig. 6. Fig. 6(a) shows that for different commanded amplitudes, two measurements of the actual amplitude closely agree with the command amplitude. The "interferometer" measurements were made by monitoring the interferometer signal directly, whereas the "controller" measurements were inferred from the controller signal x'' of Fig. 1(a).

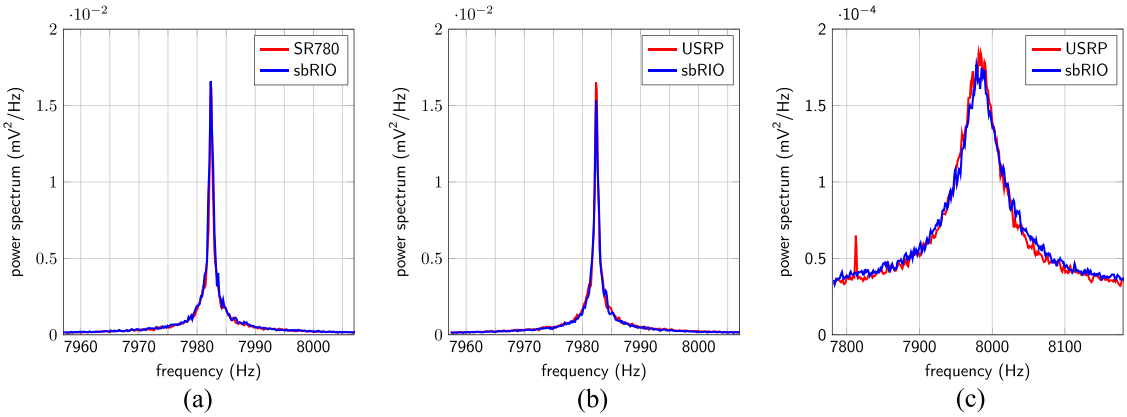

FIG. 5. Microcantilever interferometer signal power spectra comparing sbRIO microprocessor-FPGA device performance with that of (a) the SR780 spectrum analyzer and ((b) and (c)) the URSP controller.

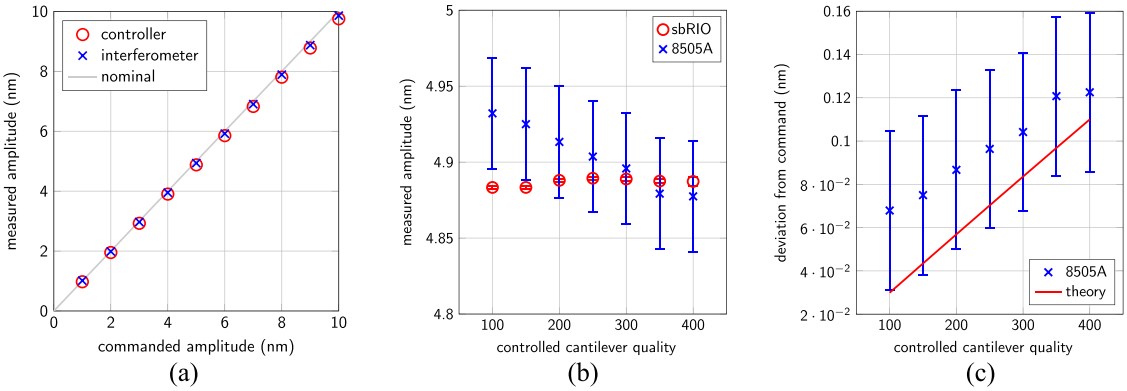

FIG. 6. Microcantilever commanded amplitude of oscillation control performance. Error bars are ±1 standard deviation. (a) Comparing commanded amplitudes to amplitudes inferred from the controller and the interferometer. (b) Comparing amplitude measurements made by the Fluke 8085A and the microprocessor-FPGA sbRIO. (c) Comparing measured to theoretical deviations from commanded amplitudes.

For a commanded amplitude of 5 nm, Fig. 6(b) shows that for different controlled qualities of the microcantilever response, two measurements closely agree with the commanded amplitude. The amplitude error bias is explained by the theoretical prediction of Eq. (10) shown in Fig. 6(c).

VI. CONCLUSIONS

From Sec. V, it is clear that the performance of MRFM instrumentation and control provided by the single-board device was competitive with (more expensive) systems much more difficult to implement and maintain. This significantly lowers the barrier for beginning and maintaining an MRFM laboratory. Several control and instrumentation implementation considerations were described in Secs. II and III; these were the key techniques instantiated by the apparatus described in Sec. IV. The focus of the apparatus description is on the software architecture that implements control and instrumentation in a single-board microprocessor-FPGA device. These techniques and architecture can be realized in several different microprocessor-FPGA devices.

ACKNOWLEDGMENTS

This work was partially supported by the Army Research Office (ARO) MURI program under Contract No. W911NF-05-1-0403.

APPENDIX A: SUPPLEMENT ON THE FPGA SOFTWARE ARCHITECTURE

More details of the software architecture are included here to assist researchers attempting to reproduce the system. Pseudocode for the FPGA portion of the controller can be found in Fig. 7. The numbering of the following list corresponds to the blocks in the flowchart of Fig. 3. The line numbers, in what follows, refer to the pseudocode of Fig. 7.

1. The FPGA code is started by the processor once all constants have been calculated and sent to the FPGA.
2. To be used to determine when to pulse the rf signals, the iterator `j` is initialized on line 1.
3. On line 3, the FPGA waits until an external trigger TTL signal goes high. This is used to trigger the FPGA to begin the next period of its sine wave generation for heterodyning. This is necessary to make sure the FPGA timing does not drift.
4. On line 4, the `DIO1` pin is set high if `j` is a multiple of N. If this pin is set high, the rf signals are enabled, and this effectively enables the rf signals every `N`th period of the cantilever period. Some MRFM experiments require this for heat reduction.
5. The iterator `i` is initialized on line 5 to count control loop cycles since the start of the cantilever period.

```
1  j = 0
2  while True:
3    if DigitalRead(DIO0)==1:
4      DigitalWrite(DIO1,mod(j,N)==0)
5      i = 0
6      while i <= samples/period:
7        looptimer.wait()
8        if i > samples/period/2:
9          DigitalWrite(DIO1,0)
10       else:
11         DigitalWrite(DIO1,mod(j,N)==0)
12       s = sine_lookup(i)
13       c = cosine_lookup(i)
14       s2 = 4.316e-5*s
15       c2 = 4.316e-5*c
16       Vin = AnalogRead(AIO1)
17       x = c2*Vin
18       y = -s2*Vin
19       xp[i] = x + X
20       yp[i] = y
21       xpp[i] = b0*xp[i] + b1*xp[i-1]
22         - a1*xpp[i-1]
23       ypp[i] = b0*yp[i] + b1*yp[i-1]
24         - a1*ypp[i-1]
25       xppp = xpp[i]*cp - ypp[i]*sp
26       yppp = ypp[i]*cp + xpp[i]*sp
27       u = c2*xppp + s2*yppp
28       AnalogWrite(AIO0,u)
29       if mod(i,dn)==0:
30         fifo.write(Xppp)
31         fifo.write(Yppp)
32       i++
33     j++
```

FIG. 7. Pseudocode for the FPGA software.

6. On line 6, a while loop commences and executes for a single cantilever period. Inside the loop, the interferometer measurement signal is sampled and mixed down, then control is applied and the resulting control signal mixed-up and applied.
7. Since `j` is used to store the number of cantilever periods completed, it must be incremented once each period is over, as on line 33.
8. The FPGA control loop is timed using a special timer function `looptimer.wait` that waits a given amount of time since it was last called. If more than the given time has elapsed since the last call, it immediately exits without waiting. This function is executed on line 7.
9. For creating the rf enable signal, it is necessary to know when half of the cantilever period has elapsed. In line 8, a test is executed to determine in which half of the cantilever period the loop is.
10. If the current time is greater than half of a cantilever period, then `DIO1` is set low on line 9.
11. If the current time is less than half of a cantilever period, then `DIO1` is set high on line 11.
12. With reference to lines 12 and 13, sine `s` and cosine `c` values are calculated for heterodyning using a lookup table function `sine_lookup` containing 1024 values. First, the number of periods per sample, a fixed point variable from zero to one, is converted to a boolean array, then to a 32-bit unsigned integer pps. This effectively multiplies it by approximately 4.3×10^9, the maximum value of a 32-bit unsigned integer. To create the sine, i is multiplied by pps and divided down to a maximum of 1024 to match the length of the stored array using a logical shift of −22 bits. The correct sine value is then retrieved from the stored array. For the cosine lookup function `cosine_lookup`, one quarter of the maximum value of a 32 bit unsigned integer is added before the logical shift. The elements stored in the array are 16 bit signed integers which range from −32 768 to 32 767. The final sinusoidal values assigned to `s` and `c` are scaled by 2^{16} to maximize precision.
13. The sine `s` and cosine `c` values must be scaled by a factor of $\sqrt{2}$ to normalize them for heterodyne control. The sine and cosine vary from −32 768 to 32 767, so the values are first cast to fixed point numbers, then multiplied by 4.316×10^{-5}. The number 4.316×10^{-5} is found using the desired normalization factor and the previous scaling, $\frac{\sqrt{2}}{2^{16}}$. The results are assigned to `s2` and `c2` in lines 14 and 15.
14. On line 16, the interferometer signal `Vin` is sampled from analog input channel `AIO1`.
15. On lines 17 and 18, the heterodyne mixdown signals are created by multiplying the interferometer signal with the cosine and sine and assigned to `x` and `y`, which correspond to x and y.
16. In lines 19 and 20, the desired commanded amplitude `X` is inserted into the control loop via summation with the in-phase signal, as described in Sec. II. The resulting expressions are assigned to the arrays `xp` and `yp`, which correspond to x' and y'.
17. The control filter H is applied to each channel to generate the control signal and filter out high frequency data. The filter constants `a1`, `b0`, and `b1` are computed previously by the processor from the analysis of Sec. II and are equal to the controller constants a_i and b_i described there. These calculations occur in lines 21–24 and are assigned to the `i` index of the `xpp` and `ypp` arrays, which correspond to x'' and y''.
18. The latencies of reads, writes, and calculation in the sbRIO and other instruments in the control loop require phase lead compensation by phase ϕ. The cosine and sine of ϕ are computed by the processor, previously, and assigned to `cp` and `sp`, respectively. From Eq. (5), the phase compensation is introduced in lines 25 and 26 to produce the mixed-down control signals `xppp` and `yppp`, corresponding to x''' and y'''.
19. Data are sent to the processor at regular intervals for display in the user interface and later analysis. If `i` is a multiple of some decimation number `dn`, then the data will be sent to the processor using the code on lines 29–31.
20. The control signal `u` (corresponding to u) is created by heterodyne upconversion: both channels are multiplied by the same sinusoid from the downconversion and added together, as shown in line 27.
21. The controller output is written to the analog output channel `AIO0` in line 28.
22. Finally, `i` is incremented.

APPENDIX B: CODE REPOSITORY

A permanent, evolving code repository is available at github.com/ricopicone/heterodyne-oscillator-controller. The

code is open-source and distributed under the GNU General Public License version 3. The authors encourage participation in the open-source project, which can be adapted to many applications that require the control of harmonic oscillators.

[1] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Desktop Edition (Basic Books, 2013), Vol. I.
[2] T. Quinn, Nature **408**, 919 (2000).
[3] E. Fischbach and C. Talmadge, *The Search for Non-Newtonian Gravity* (Springer, New York, 2012).
[4] M. W. Moore, J. H. Steffen, and P. E. Boynton, Rev. Sci. Instrum. **76**, 085106 (2005).
[5] N. Jalili and K. Laxminarayana, Mechatronics **14**, 907 (2004).
[6] G. Binnig and H. Rohrer, Surf. Sci. Lett. **152**, A106 (1985).
[7] J. A. Sidles, Appl. Phys. Lett. **58**, 2854 (1991).
[8] S. Kuehn, S. A. Hickman, and J. A. Marohn, J. Chem. Phys. **128**, 052208 (2008).
[9] M. Poggio and C. L. Degen, Nanotechnology **21**, 342001 (2010).
[10] C. L. Degen, M. Poggio, H. J. Mamin, C. T. Rettner, and D. Rugar, Proc. Natl. Acad. Sci. U. S. A. **106**, 1313 (2009).
[11] J. G. Longenecker, H. J. Mamin, A. W. Senko, L. Chen, C. T. Rettner, D. Rugar, and J. A. Marohn, ACS Nano **6**, 9637 (2012).
[12] C. E. Issac, C. M. Gleave, P. T. Nasr, H. L. Nguyen, E. A. Curley, J. L. Yoder, E. W. Moore, L. Chen, and J. A. Marohn, Phys. Chem. Chem. Phys. **18**, 8806 (2016).
[13] R. A. Picone, "Separative magnetization transport: Theory, model, and experiment," Ph.D. thesis, University of Washington, 2014.
[14] R. A. Picone, J. L. Garbini, and J. A. Sidles, J. Magn. Magn. Mater. **374**, 440 (2015).
[15] J. M. Nichol, E. R. Hemesath, L. J. Lauhon, and R. Budakian, Phys. Rev. B **85**, 054414 (2012).
[16] J. L. Garbini, K. J. Bruland, W. M. Dougherty, and J. A. Sidles, J. Appl. Phys. **80**, 1951 (1996).
[17] J. P. Jacky, J. L. Garbini, M. Ettus, and J. A. Sidles, Rev. Sci. Instrum. **79**, 123705 (2008).
[18] T. E. Kriewall, J. L. Garbini, J. A. Sidles, and J. P. Jacky, J. Dyn. Syst., Meas., Control **128**, 577 (2005).
[19] D. Weitekamp, "Radiative reduction of entropy," U.S. patent 6,841,995 (11 January 2005).
[20] Only minor adjustments are required to apply the methods described here to other MRFM apparatuses.
[21] J. A. Sidles, J. L. Garbini, K. J. Bruland, D. Rugar, O. Züger, S. Hoen, and C. S. Yannoni, Rev. Mod. Phys. **67**, 249 (1995).
[22] L. Gross, F. Mohn, N. Moll, P. Liljeroth, and G. Meyer, Science **325**, 1110 (2009).
[23] See the work of Jacky *et al.*[17] for details on α and β.
[24] Note that a typographical error appears in the definitions of gain and cutoff frequency in the work of Kriewall *et al.*,[18] Eq. (22), but is rectified in the present work.
[25] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*, World Student Series (Addison-Wesley, 1998).
[26] See the work of Jacky *et al.*[17] for details.
[27] We have made the assumption that the heterodyne frequency is identical to the microcantilever resonance frequency, which is valid, assuming the heterodyne system is provided an accurate measurement of the microcantilever resonance frequency.
[28] The phase adjustment is useful for control, but is inconsequential for measurement, which only changes in phase matter. Therefore, the phase is not typically "un-compensated." The commanded amplitude can easily be subtracted from the data, although it is usually inconsequential because experiments with the commanded amplitude typically measure only phase shifts.
[29] S. Davis, "Control of microcantilevers using a stand-alone FPGA-processor platform," M.S. thesis, University of Washington, 2013.
[30] H. J. Mamin and D. Rugar, Appl. Phys. Lett. **79**, 3358 (2001).