2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Low Latency Demodulation for Atomic Force Microscopes, Part I Efficient Real-Time Integration

Daniel Y. Abramovitch

Abstract— **This paper describes methods for doing high-speed, low latency, coherent demodulation of signals for dynamic or AC mode in Atomic Force Microscopes (AFMs) [1]. These demodulation methods allow the system to extract signal information in as little as one cycle of the fundamental oscillation frequency. By having so little latency, the demodulator minimizes the time delay in the servo loop for an AC mode AFM. This in turn minimizes the negative phase effects of the demodulation allowing for higher speed scanning. This part of the paper describes the mixing and integration portion of the demodulator. Part II [2] describes efficient methods for extracting magnitude and phase in real time.**

I. Introduction

Dynamic mode AFM, which involves an oscillation of the cantilever in the proximity of the surface at a frequency close to the resonant frequency of the cantilever is depicted in Figure 1. In non-contact mode, the amplitude of the oscillation is slightly less than the nominal tip/surface distance so that while there is interaction between the tip and surface, this never enters into what would be considered contact. In the most common form of dynamic mode, also known as AC mode, or intermittent contact mode [3], [4], the amplitude of the free oscillation is slightly larger than the nominal tip/surface distance. When the tip comes into proximity with the surface, the oscillation amplitude, phase, and frequency are modulated. By detecting this modulation and closing a feedback loop on the amplitude of the oscillation, this amplitude can be maintained at a constant level (modulo the bandwidth of the system). Typically, the control signal represents the surface topography.

Dynamic mode imaging is done using cantilevers of various frequency ranges which are described in [5]. Often as the cantilever resonant frequency goes up, they get stiffer and have a higher Q. The higher Q provides greater amplitude amplification of the drive signal and better frequency discrimination for small shifts due to surface interaction. However, the extra stiffness of the cantilever might damage some materials, so there is a trade-off to be made on increasing the cantilever resonance. Because dynamic mode produces lower sheer forces on the sample than contact mode, the imaging of biological samples, is often done using this technique.

Although dynamic mode operation of AFMs is favored for imaging of soft samples, this operation is hampered by its slow speed. There are several reasons for this, as described in [5]:

D. Y. Abramovitch is a senior research engineer in the Nanotechnology Group at Agilent Laboratories, 5301 Stevens Creek Blvd., M/S: 4U-SB, Santa Clara, CA 95051 USA, danny@agilent.com

- The Q factor of the cantilever affects the time response. The cantilever is usually oscillated near its resonant frequency to get reasonable deflection amplitudes with low levels of input signal. Due to nonlinear interactions with the surface, the tip oscillation amplitude responds almost instantaneously to a step up in the surface. However, when there is a step down in the surface height, the response time of the cantilever oscillation will be proportional to Q/ω_o, where ω_o is its resonant frequency [6]. The flywheel action (which the author likes to call the *Wiley E. Coyote effect*), also introduces a limitation on the imaging speed without imaging artifacts.
- In AC mode, information about the surface is only available during the contact interval, which happens once every period of the oscillation. Consequently, the duty cycle of tip/surface interaction is considerably reduced as compared with contact mode.
- The surface information of interest is typically at a frequency well below that of the oscillation frequency and must be extracted from the oscillatory return signal via demodulation.

It is the role of the demodulator to extract surface information from the return signal. Typically there is a trade-off between the fidelity of the extracted information and shortening the demodulation time. However, speeding up dynamic mode operation depends upon having a high fidelity, low latency demodulator. Consider driving the cantilever with a sine wave:

$$d(t) = D_0 \sin(\omega_0 t). \tag{1}$$

The signal, $s(t)$, from the optical sensor, is composed of harmonics of $\sin(\omega_0 t)$, i.e.

$$s(t) = A_0 + \sum_{k=1}^{\infty} \left(A_k \sin(k\omega_0 t) + B_k \cos(k\omega_0 t) \right). \tag{2}$$

We can expect that if the drive signal is large enough and the tip/surface interaction is set to be a small fraction of the free space oscillation, then the majority of the signal will be dominated by the first harmonic,

$$s(t) \approx A_1 \sin(\omega_0 t) + B_1 \cos(\omega_0 t) = C_1 \sin(\omega_0 t + \phi_1). \tag{3}$$

where

$$C_1 = \sqrt{A_1^2 + B_1^2} \text{ and } \phi_1 = \arctan \frac{A_1}{B_1}. \tag{4}$$

Because dynamic mode typically operates near the cantilever resonance [7], there is a relationship between the

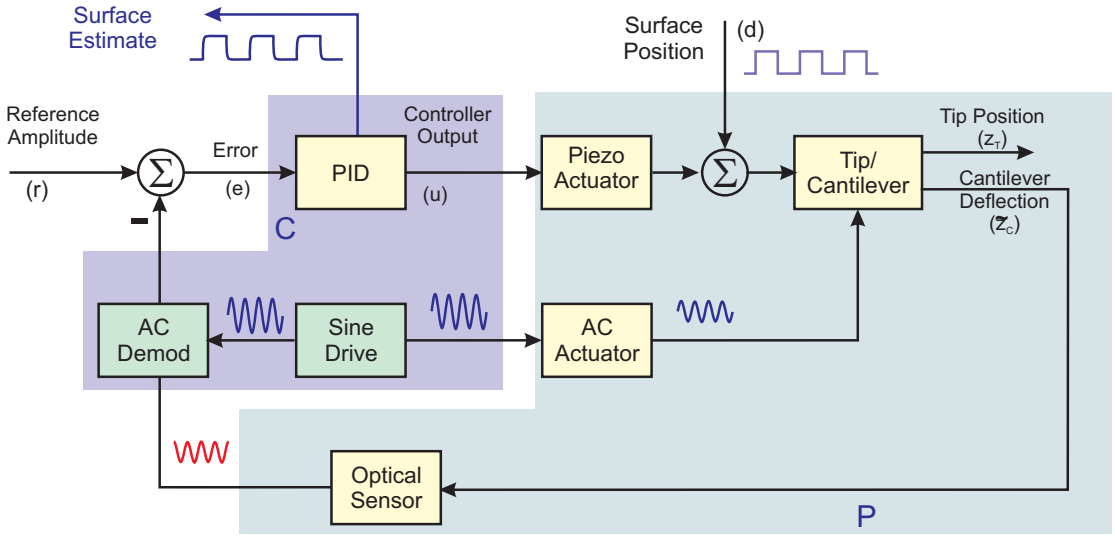

Fig. 1. An AFM Control Block Diagram in dynamic mode.

amplitude shift, phase shift, and frequency shift seen due to the surface/tip interaction. Thus, both the imaging and the Z-axis servo loop can be driven by one of several demodulated signals.

- **Amplitude Modulation (AM):** In this mode, the change in the amplitude of the oscillation is detected and used as the error signal for the feedback loop. The speed of AM-AFM is often limited by the high Q-factor of the cantilever.
- **Phase Modulation (PM):** In this mode, the change in the phase difference between the cantilever drive and the returned deflection signal is detected. While feedback on the amplitude is easier to implement, the phase signal can be used to measure other surface properties like energy dissipation [8].
- **Frequency Modulation (FM):** In this mode, the change in the oscillation frequency of the returned deflection signal is detected. FM-AFM typically requires extremely high-Q cantilevers so that the frequency shift can be detected. This has meant that FM-AFM is most often done in a vacuum where the lack of air damping makes the cantilever Q seem much larger.

Non-coherent, or non-synchronous demodulation, using an analog RMS-to-DC circuit (or its digital equivalent) can extract the signal magnitude, but not the phase. Both magnitude and phase can be extracted using a lock-in amplifier, which is a synchronous device that mixes in-phase and quadrature signals with the input signal and then integrates for a known period of time. Typical external lock-in amplifiers are slow, since tracking speed is often secondary to accuracy. Like the RMS-to-DC circuit, they often integrate over at least 10 periods of the input signal. For example, the 36 ms settling time of the AD736 [9] is 3,168 periods of the 88 kHz signal used in the examples of Section III. There has been a push among AFM manufacturers to include digital lock-in amplifiers in their AFM controllers. To be useful for high speed control, however, these lock-in amplifiers need to both have low latency and high fidelity. This is demonstrated in this paper.

II. Coherent Demodulation for AFMs

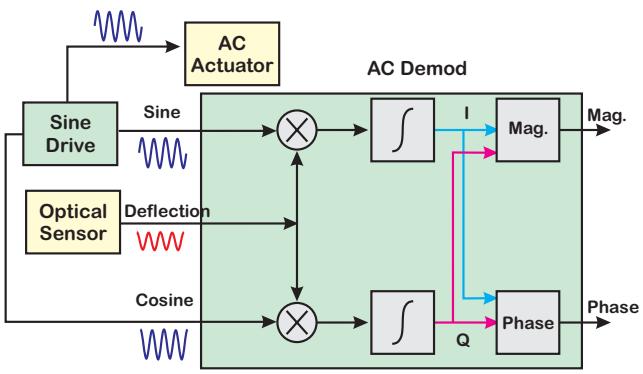

Fig. 2. Coherent demodulation for AFM.

In coherent demodulation, the signal to be demodulated is separately mixed with an in-phase and a quadrature (90° out of phase) signal and then integrated. If

$$s(t) = C_1 \sin(\omega_0 t + \phi_1) + n(t), \text{ then} \tag{5}$$

$$I(t) = s(t)\sin(\omega_0 t) \text{ and} \tag{6}$$

$$Q(t) = s(t)\cos(\omega_0 t), \tag{7}$$

where $n(t)$ is the noise in $s(t)$. The basic principle of coherent demodulation is based on the idea that if one sets the mixing signal to the same fundamental period as the drive signal, $T_0 = \frac{1}{f_0} = \frac{2\pi}{\omega_0}$, and sets the integration period, T equal to a positive integer number of those periods, MT_0, then most of the terms in the integrals drop out. In particular,

we can set M as low as 1 and

$$\frac{1}{MT_0}\int_0^{MT_0} I(t)dt = \frac{C_1}{2}\left(\cos(\phi_1)\frac{1}{MT_0}\int_0^{MT_0} dt - \frac{1}{MT_0}\int_0^{MT_0}\cos(2\omega_0 t+\phi_1)dt + \frac{1}{MT_0}\int_0^{MT_0} n(t)\sin(\omega_0 t+\phi_1)dt\right). \quad (8)$$

has the properties that the second term on the right hand side goes to 0 for all positive M. The third term goes to 0 for increasing MT_0 as long as $n(t)$ is uncorrelated with the mixing sinusoids.

Such precise control of the integration period is hard to do in an analog circuit but completely doable in a digital operation. A similar result for $Q(t)$ is obtained from integrating Equation 7 in a similar fashion.

As MT_O gets large the contribution of $n(t)$ goes to 0, yielding the familiar relationships

$$\frac{1}{MT_0}\int_0^{MT_0} I(t)dt \approx \frac{C_1}{2}\cos(\phi_1) \text{ and} \quad (9)$$

$$\frac{1}{MT_0}\int_0^{MT_0} Q(t)dt \approx \frac{C_1}{2}\sin(\phi_1). \quad (10)$$

A. Discrete Approximation of the Integral

There are several issue with standard methods of demodulation. The first is that imperfections in the integration approximation and noise in the signal require that MT_O be large, relative to the period of the frequency at which demodulation is to take place. However, for control of high speed systems in general, and AFMs in particular, this long integration period results in extra delay in the system, and this delay translates into negative phase which in turn limits bandwidth. In order to minimize this latency, we need to make the integral more accurate for short integration times and minimize the noise, $n(t)$, in the return signal.

The second is that for the reasons above we wish to use digital methods. To be generally useful, a digital demodulator needs to be functional over the entire frequency span of cantilever resonant frequencies. To integrate a sampled signal over an integer number of periods of oscillation of the signal,

$$NT_S = MT_0, \quad (11)$$

where N are the number of samples in the integration, T_S is the sample period, M is the number of periods of oscillation, and T_0 is the period of oscillation. However, as the data sample rate is rarely an integral multiple of the oscillation frequency, it is difficult to make Equation 11 hold. Most digital systems are run at a fixed sample rate, $f_S = \frac{1}{T_S}$. The oscillation frequency, f_0, in an AFM is set by the cantilever oscillation frequency. That is to say, f_0 might change slightly, but f_S will not.

The adjustments to T_0 to make equality hold in Equation 11 can be kept small if N and M are made large. While this approach may be feasible for an offline measurement, or for producing signal processing results that are not used in the feedback loop, this method will add to latency in the integral calculation. A more practical solution is to find the minimum N such that

$$NT_S \le MT_0 \le (N+1)T_S. \quad (12)$$

In the most common case when equality does not hold, the last period of the integration is a partial one, as shown in Figure 3. This will require $N+1$ samples where the first N samples of the integral integrate over the complete sample period and the last one is interpolated over a partial sample.

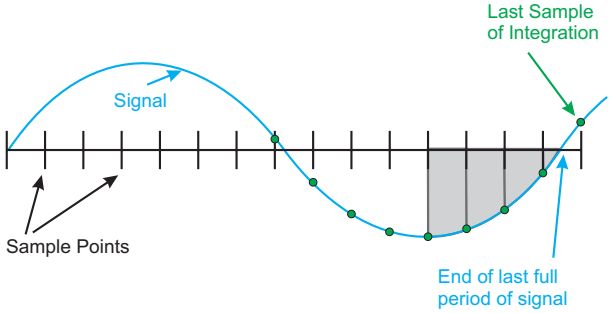

Fig. 3. Integrating the partial sample of a sampled sinusoid.

Digital quadrature is documented in many numerical computation texts [10], [11]. Generally, the algorithms for quadrature will make use of a polynomial fit over some number of sample points to approximate the function. The fit of a L^{th} order polynomial will involve $L+1$ points. In applications where latency (time delay) is not an issue, one can achieve higher accuracy by conducting the integral between samples k and $k+1$ using samples on either side of this interval.

For our problem, we want to minimize the latency of the integral and for this the simplest discrete integral approximations are the forward and backward rectangular rule approximations, and the trapezoidal rule approximation. Because of their small amount of delay, these are often used in generating discrete equivalents of analog controllers [12]. The forward rectangular rule has a single period delay. The backwards rectangular rule has zero delay. Finally, the trapezoidal rule has a half sample period delay.

Standard practice in digital lock-in amplifiers is to use one of the rectangular rule approximations and rely on using many periods of oscillation to drive the error to 0. However, by using a higher-order approximation and a partial sample integral, we can cut the error down with far fewer periods of integration.

B. Practical Implementation of the Discrete Integration

In this paper, a trapezoidal rule integration is used. This seems to provide a reasonable compromise between minimizing latency and integration accuracy. consider the trapezoidal rule implementation of our integral:

$$\int_{x_0}^{x_N} y(t)dt \approx \sum_{k=0}^{N-1}\left(\frac{y_{k+1}+y_k}{2}\right)T_S, \quad (13)$$

where T_S and N are defined as in Equation 12. Between N and $N+1$, we will have a partial interval integral that must be computed

$$\int_{x_N}^{x_k+T_S h} y(t)dt \approx \left(\frac{y_{N+1}+y_N}{2}\right) hT_S, \tag{14}$$

where $0 \leq h \leq 1$ and

$$h = \frac{MT_0 - NT_S}{T_S}. \tag{15}$$

Note that hT_S is the integration time needed to complete the M^{th} period of oscillations at f_0, so the fraction of a sample period that this represents is given by h. Putting these together and looking back in time rather than forward, we get

$$\int_{kT_S - MT_0}^{kT_S} y(t)dt \approx S_k \text{ where} \tag{16}$$

$$\frac{S_k}{T_S} = \sum_{j=0}^{N-1} \left(\frac{y_{k-j}+y_{k-(j+1)}}{2}\right) + \left(\frac{y_{k-N}+y_{k-(N+1)}}{2}\right) h. \tag{17}$$

We now need to focus on the properties of the sum, S_k and how to compute this efficiently. We see that S_k can be rewritten as

$$\frac{S_k}{T_S} = \frac{y_k}{2} + \sum_{j=0}^{N-1} y_{k-j} + \frac{y_{k-N}}{2} + h\left(\frac{y_{k-N}+y_{k-(N+1)}}{2}\right). \tag{18}$$

Equation 18 is very instructive because it shows us that the integral can be simply constructed as a FIR filter. We can factor out a single scale factor, $\frac{T_S}{2}$, and then we have a main integral corresponding to the terms before the term scaled by h and the fractional portion, scaled by h. It is also instructive that very little about this formula is dependent upon the sample interval, f_S, and the oscillation frequency, f_0. Basically, a change in f_0, f_S, and/or the number of periods in the integral, M, changes only N and h. For a given M, T_0, and T_S, we pick N from Equation 12 and h from Equation 15.

So, we see that this integration problem can be put in the general form of an FIR filter, providing we can handle the bookkeeping for the computation. At each time step, new data comes into one side of the FIR and old data is discarded from the other side. Actually doing this in real-time hardware presents three issues:

- First, shifting all the values of $y(j)$ back one step in time can be expensive in terms of computation time. At each time step, N values have to be shifted in their memory locations so that they line up with the coefficients of the filter at the next time step.
- Second, the longer the filter, the longer the number of computations. Varying latency is a problem for feedback systems and must be avoided.
- Third, any new frequency requires loading a new set of $N+1$ coefficients.

In wanting a method that is efficient in any possible architecture, it is worth looking at an incremental or iterative method, that is, a method that uses a running sum for the integral and only makes adjustments to this running sum at each time step.

Advancing Equation 18, forward one step in time, and taking the difference with the current value yields

$$\begin{aligned} S_{k+1} - S_k \\ = \quad & \frac{T_S}{2}\Bigg[y_{k+1} - y_k + 2\sum_{j=1}^{N-1} y_{k+1-j} - 2\sum_{j=1}^{N-1} y_{k-j} \\ + \quad & y_{k-(N-1)} - y_{k-N} + h\left(y_{k-N} - y_{k-(N+1)}\right)\Bigg] \end{aligned} \tag{19}$$

which – through algebra that would result in extra page charges – reduces to

$$\begin{aligned} \Delta S_{k+1} = S_{k+1} - S_k = \frac{T_S}{2}\big[y_{k+1} + y_k \\ + y_{k-(N-1)} - y_{k-N} + h\left(y_{k-N} - y_{k-(N+1)}\right)\big]. \end{aligned} \tag{20}$$

This is a wonderful result, because Equation 20 has a form that uses a small, fixed number of terms. Even if higher-order integration methods are used, this form would have a larger, but fixed number of terms, independent of N. We need to keep track of old values of the integral, but they are used sparingly in the calculation. If we start with $S_0 = 0$, then we can compute S_k from

$$S_k = \Delta S_k + S_{k-1}. \tag{21}$$

In short, we have implemented a quadrature integration as an FIR, and compute the values of that FIR recursively as if it were an IIR. What keeps this an FIR is that at a later time we subtract off exactly the same value that was added to the cumulative sum. We still need to keep track of a potentially large set of prior sample values. If we were to move each sample back one time step in memory, this would create large amounts of bookkeeping computations. Instead, we can use circular addressing of a memory buffer of old sample values as is often done in DSP calculations. New data is written into the memory and old data is read out of it. The memory addresses at which this happens are computed using circular addressing, with the indices of these addresses moving at each time step. Thus, the update of the memory buffer at any one time step requires reading the oldest value in the filter from memory and writing the newest value into memory. This can be done as easily in a FPGA as in a processor and is illustrated in Figure 4.

In Figure 4 we see that we can use a single large memory block and put the filter in part of it. The use of a larger block allows the filter length to be quite flexible, so that it is easy for the user to change oscillation frequency or the number of periods of oscillation that are needed for the calculation. Changing these simply results in a change in the size of the memory used in the block. We see here that for the N chosen as above, we only have $N+2$ words of memory allocated for the filter. At any time k, a new sample will

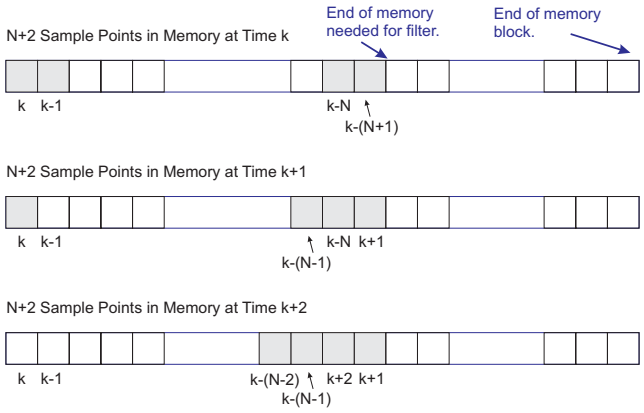

Fig. 4. Circular addressing of filter memory

be written at a memory location denoted by k, and an old sample will be read from memory at location $k-(N+1)$. When the next sample comes in it should go to the left of sample k, but since our drawing shows this at the beginning of the memory block, the writing index is chosen to be at the end of the block, where the no longer useful data from $k-(N+1)$ is held. Thus, the sample from $k+1$ is written there, while the index for reading decrements so as to point to location $k-N$. We can see that this process can go on easily, simply by initializing the indices in the right locations and then moving them in synchrony together.

III. Examples

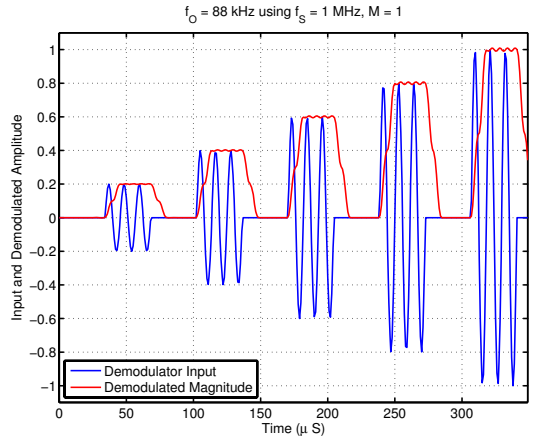

Fig. 5. Matlab simulation demonstrating speed of convergence.

Equations 20 and 21 are easy to implement simply in real-time processors such as DSPs or FPGAs. In this section are simulation examples to illustrate the behavior of the integration portion of the demodulator. Similar results are shown in Part II [2] to illustrate the extraction of magnitude and phase from the integrated quantities.

An illustration of the convergence of the integrator is shown in the simple Matlab simulation of Figure 5. The input signal is at 88 kHz and is switched "on" and "off" with the magnitude stepped up with each "on" period. The demodulated magnitude is plotted on the same axis as the input signal. As predicted by the analysis, the integrator

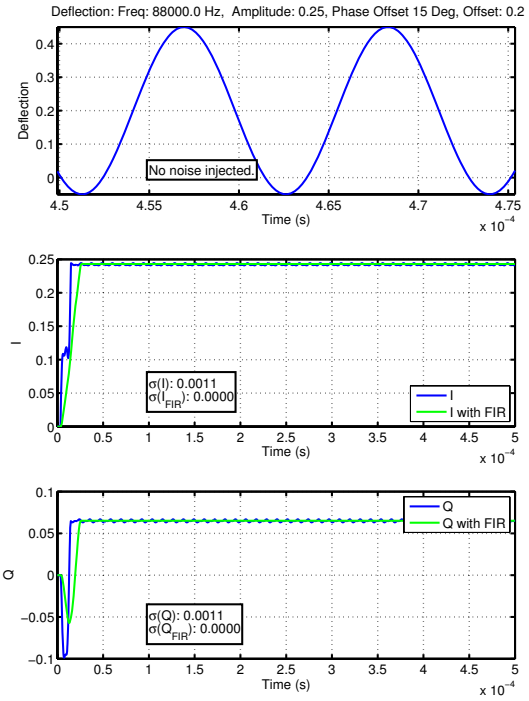

Fig. 6. Output of ModelSim Simulation of FPGA based demodulator. The oscillation frequency is 88 kHz. The normalized deflection amplitude is 0.25. There is a normalized offset of 0.2 in the signal level, and the phase of the signal driving the deflection is 15° ahead of that of the in-phase (sine) mixing signal at the beginning of the simulation.

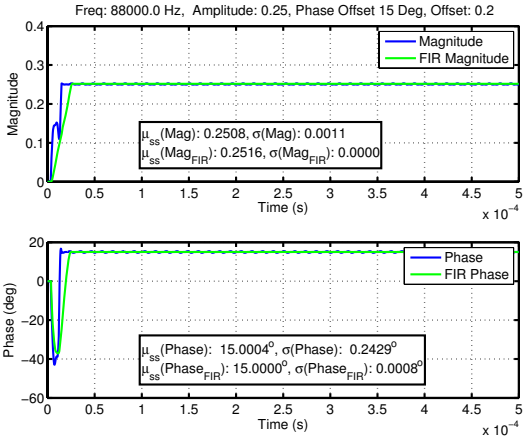

Fig. 7. Floating point magnitude and phase extracted from the simulation in Figure 6.

yields I and Q in 1 integration period ($M = 1$), which is mapped to magnitude in the plot.

The demodulator architecture, implemented in FPGA hardware, was simulated using ModelSim 6.6b [13], and signals were normalized back to real numbers. The signals of interest to this paper were saved to an ASCII file, which was processed in Matlab. The last 20% of the data was used to compute the steady state averages (μ) and standard deviations (σ) of these signals. The means for the I and Q signal do not mean much to the reader, but the small σ for these signals does show the accuracy of the integrator. In the case of the magnitude and phase μs and σs, the values can be compared to the original inputs. Note that the σ value for phase is in

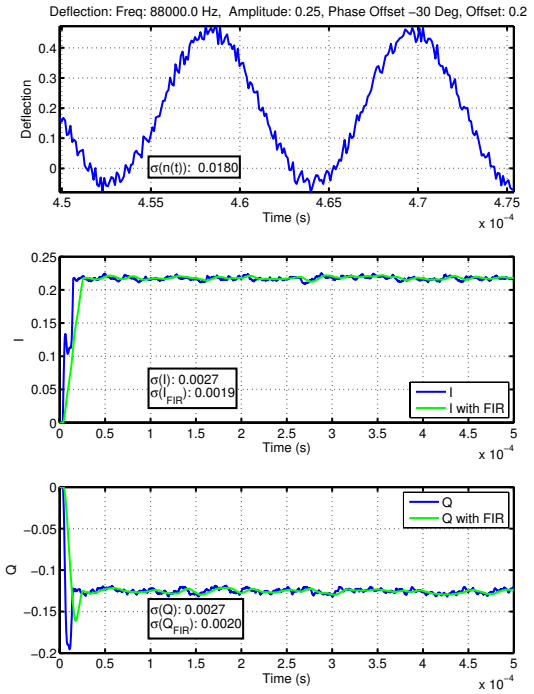

Fig. 8. Output of ModelSim Simulation of FPGA based demodulator. The oscillation frequency is 88 kHz. The normalized deflection amplitude is 0.25. There is a normalized offset of 0.2 in the signal level, and the phase of the signal driving the deflection is 30° behind of that of the in-phase (sine) mixing signal at the beginning of the simulation.

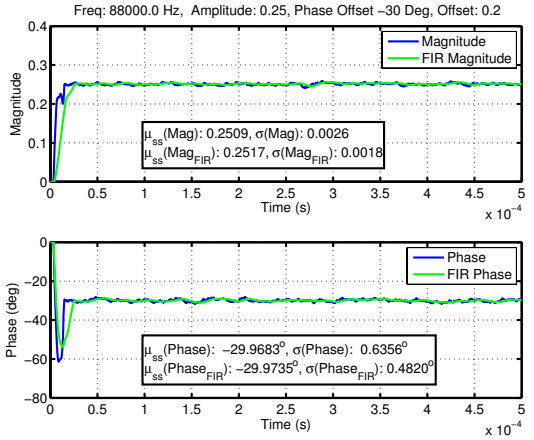

Fig. 9. Floating point magnitude and phase extracted from the simulation in Figure 8

degrees.

Figures 6 –9 show the results of the simulator when driven with an 88 kHz signal, which had an amplitude of 0.25, an offset of 0.2 and a phase advance (as compared to the in-phase mixing signal) of 15° (in Figures 6 and 7) or a delay of 30° (in Figures 8 and 9). The top plots of Figures 6 and 8 are zoomed in to better show the effects of noise on the signal. The lower two plots show the results of the I and Q branch integrations. The in-phase (I) and quadrature (Q) branches converge quickly, although small imperfections in the integration result in some ripple in these signals. Passing these signals through the FIR filter described in Part II [2], results in the removal of these effects. With no noise injected, the standard deviation (σ) is minuscule. The rejection of white noise while using a single integration period is small. The rejection can be increased by integrating over multiple oscillation periods, but at the cost of more latency.

While the simulation accurately simulates synthesizable blocks of the FPGA, it can also simulate blocks that cannot be put into logic. Thus, Figure 7 and 9 show real values of magnitude and phase extracted from the integrator outputs. Note that with no noise injected, the deviation (σ) of the magnitude and phase from their steady state values (μ) is extremely small (and virtually non-existent after the application of the FIR). The effect of signal noise, $n(t)$, is diminished despite the short integration time.

IV. Conclusions

This paper demonstrates a low latency, high accuracy, AC mode demodulator that is applicable for high speed, real-time applications. The algorithm implements classical coherent demodulation theory in a clean and flexible structure with minimal computational overhead. This part of the paper has described the mixing and integration portion of the demodulator. Part II [2] describes efficient methods for extracting magnitude and phase in synthesizable blocks.

References

[1] D. Y. Abramovitch, "Coherent demodulation with reduced latency adapted for use in scanning probe microscopes," United States Patent 7,843,627, Agilent Technologies, Santa Clara, CA USA, November 30 2010.

[2] D. Y. Abramovitch, "Low latency demodulation for atomic force microscopes, Part II: Efficient calculation of magnitude and phase," in *Proceedings of the IFAC 18th World Congress*, (Milan, Italy), IFAC, IFAC, August 28–September 2 2011.

[3] Q. Zhong, D. Inniss, K. Kjoller, and V. Ellings, "Fractured polymer/silica fiber surface studied by tapping mode atomic force microscopy," *Surf. S. Lett.*, vol. 290, pp. L688–L692, Jun. 1993.

[4] V. B. Elings and J. A. Gurley, "Jumping probe microscope," United States Patent 5,266,801, Digital Instruments, Santa Barbara, CA USA, November 30 1993.

[5] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, "A tutorial on the mechanisms, dynamics, and control of atomic force microscopes," in *Proceedings of the 2007 American Control Conference*, (New York, NY), pp. 3488–3502, AACC, IEEE, July 11–13 2007.

[6] T. Sulchek, G. G. Yaralioglu, S. C. Minne, and C. F. Quate, "Characterization and optimization of scan speed for tapping mode atomic force microscopy," *Rev. Sci. Instrum.*, vol. 73, p. 2928, 2002.

[7] R. W. Stark, G. Schitter, and A. Stemmer, "Tuning the interaction forces in tapping mode atomic force microscopy," *Physical Review B*, vol. 68, pp. 085401–1–085401–5, 2003.

[8] B. Anczykowski, B. Gotsmann, H. Fuchs, J. P. Cleveland, and V. B. Elings, "How to measure energy dissipation in dynamic mode atomic force microscopy," *Appl. Surf. Sci.*, vol. 140, pp. 376–382, 1999.

[9] C. Kitchen and L. Counts, *RMS-to-DC Conversion Application Guide*. Analog Devices, Inc., second ed., 1986.

[10] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1988.

[11] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London: Academic Press, 1981.

[12] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*. Menlo Park, California: Addison Wesley Longman, third ed., 1998.

[13] Mentor Graphics, *ModelSim*. Mentor Graphics, www.mentor.com/products/fpga/simulation/modelsim, 2011.