# Real Time Interferometric Imaging of the Plasma Etch Process

by

Jonathan L. Claman

Bachelor of Science, Massachusetts Institute of Technology (1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994



Author .....
Department of Electrical Engineering and Computer Science
May 16, 1994

Certified by .....
Duane S. Boning
Assistant Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
F. R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

# Real Time Interferometric Imaging of the Plasma Etch Process

by

Jonathan L. Claman

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 1994, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

As microelectric device sizes continue to shrink, the monitoring and control of the fabrication processes are becoming more important for reliable yields. Traditional process parameters are pre-set, and thus can not compensate for slight variations in a given run. This thesis researches the use of a CCD camera to monitor the plasma etch process in real time. The monitor's data will be used in other research as the input for real time control.

Laser interferometry, a commonly used diagnostic, can only monitor one small part of the wafer. If one desired to look at more points on the wafer, more lasers could be used, each aimed at a different location. A CCD camera monitoring the wafer acts as a $1000 \times 1000$ array of lasers. Thus, by looking at one pixel only, the CCD interferometer can function like a laser interferometer. Or several points can be looked at, as if there were several laser interferometers. The interferometry signal can be processed to yield an etch rate and an endpoint at each pixel. Data from different parts of the wafer provide uniformity information that can be used for real time control.

This thesis implements a system capable of real time image acquisition and analysis. Algorithms to perform etch rate, endpoint, and film thickness measurements are discussed. A discussion of interferometry modeling, including non-ideal device characteristics which affect the models, is also presented. Finally, the system architecture, both hardware and software, is discussed.

Thesis Supervisor: Duane S. Boning
Title: Assistant Professor of Electrical Engineering

## Acknowledgments

I would first and foremost like to thank Professor Duane Boning for his wisdom, guidance, support, and encouragement, which helped get me through even the most difficult of times.

My officemate, Ka Shun Wong, deserves much of the credit on this research, for it was he who made many of the fundamental observations that led this research down bright new paths. In addition, he has been possibly the most responsible and easy-to-get-along-with officemate I have ever had.

I would also like to thank Professor Herb Sawin for the access to his lab equipment, and Tim Dalton for his seemingly infinite knowledge on CCD interferometry and his tolerance for Ka Shun's and my constant begging him for more data.

Thanks are also due to Professor Boning's other graduate students, Safroadu Yeboah-Amankwah, Dave White, and William Moyne, for their patience and advice during our group meetings.

Throughout all the hard work, everyone deserves a little break now and then. That break during lunch time was certainly made more enjoyable by the company of Jason Cornez and Neil Tender. Lunches would have been quick, boring, and often skipped if they weren't around. In addition, I'd like to thank Jason for his software expertise and Neil for his DSP insight and the last-minute proof reading of this thesis.

I'd also like to thank Phil Pan, Peggy Hsieh, and Victoria Parson for being there on the phone (and in the bars) to help me relax and have fun dring the busy months of research. Dave, mentioned earlier, deserves thanks here, too.

Gail Levin has, of course, helped this past semester been a lot more enjoyable for me. The trips out to Indiana helped me to keep perspective between this research and the "real" world outside of it, in addition to giving me a great excuse to put down my research for a couple of days at a time.

A very special thanks to my parents for all of their love. It was their encouragement throughout the years that has pushed me to attain such high goals. Larry and Susan have been the most wonderful brother and sister-in-law ever, from cooking me dinner to taking care of me when I was sick. My whole family is great, and I couldn't have accomplished any of this without them.

Finally, I would like to thank my sponsors, because this work would definitely not have been possible without them. This work was sponsored in part by the Advanced Research Projects Agency under Contract N00174-93-C-0035, and by the Semiconductor Research Corporation under Contract 93-MC-503.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Ever since integrated circuits were developed, there has been a constant drive towards shrinking device sizes. Feature sizes of advanced production devices were near 25$\mu$m in 1960, 5$\mu$m in 1975, and are currently near 0.5$\mu$m. This decrease of approximately 11% per year is expected to continue [1]. The drive in decreasing device sizes has been made possible through the use of increasingly sophisticated fabrication techniques, recently including process control.

A controller can be thought of as a system which makes a decision based on some input. This thesis explores a key input to a system which will control the plasma etch process, one of several critical steps required in the manufacturing of semiconductors. The input system used is a CCD camera which takes pictures of the wafer *in-situ* and in real time, i.e. during the process. This thesis is motivated by the need for process control, and thus the need for a monitoring system which will provide information to the plasma etch controller.

## 1.2 Background

Consider the act of filling a car with gas as a control example. One could use a run-by-run controller in which gas is pumped for say, 60 seconds. The tank is then examined to determine how close to full it is, or how much gas has spilled from overfilling the tank. Based on this observation, an adjustment is made in the amount of time gas is pumped at the next fillup in an attempt to compensate for the over or under fill. The goal is a gas

tank which is perfectly "topped off." This kind of control may work if the car is always operated for the same period of time and over the same routes. A better control algorithm, such as the one installed on virtually all gas pumps in this country, allows the tank to be monitored while it is being filled. This better control is fundamentally enabled by a sensor which detects when the tank is full, and provides feedback to the gas pump by shutting off the valve. This system has the advantage of not relying on uniformity in the state of the incoming car, nor on pre-determined pumping process parameters, by adapting to each gas tank individually. This thesis is about sensors which enable control in semiconductor fabrication, not pumping gas.

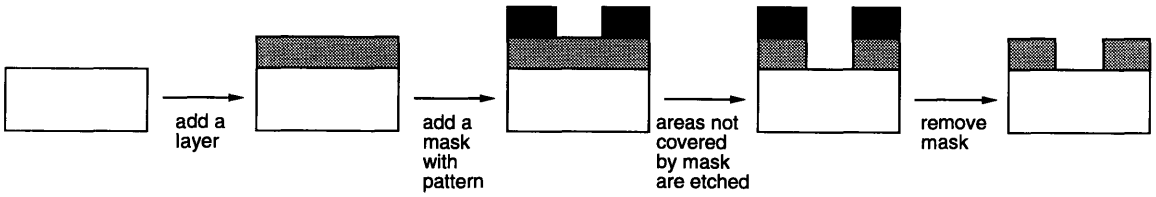


Figure 1-1: A simplified process flow for semiconductor fabrication.

Microelectric fabrication is a process of repeatedly adding a layer of material, patterning it to a desired shape, and then adding another layer. A simplified flow for this process is shown in Fig. 1-1. This thesis is concerned with the etching process. A poorly executed etch process can result in incomplete clearing of etch areas, known as "under etch" (Fig. 1-2a), in which not all of the material has been removed. The other extreme is over etch, as shown in Fig. 1-2b, where too much material has been removed. These poor etch results can potentially lead to open or short circuits, as well as other device and quality problems. Often both over etch and under etch exist across a wafer after processing, but the inaccuracies are small enough so as to not affect the device functionality. As device sizes continue to shrink, however, there is demand for more precise etching processes.

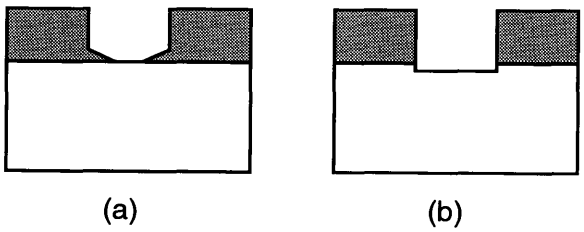


Figure 1-2: Uncontrolled etching processes can lead to (a) under etch with partial clearing or (b) over etch.

Recent work at M.I.T. has used a run-by-run control approach [2]. In this algorithm, a wafer is processed, and then measurements are performed to determine the accuracy of the previous run. Process parameters (such as gas flows, power, or magnetic field strength) may be adjusted for the next wafer based on the diagnostic results. This approach successfully compensates for run to run variations such as machine drift (which is common as the machine ages), but cannot account for such variations as wafer non-uniformities from previous process steps.

An improvement to run-by-run control is "within-a-run" control. By monitoring the wafer during the run, slight deviations from desired characteristics can be detected and corrected, always producing a wafer conforming to specifications. We are attempting to control wafer uniformity; that is, all points on the wafer, regardless of their location, should be etched at the same rate, and should reach endpoint simultaneously. The CCD camera used in this research provides the ability to examine the entire wafer during a process, making it a worthy tool for use in semiconductor process control.

## 1.3 Overview

This thesis begins by explaining the origin of the interferometry signal and model in Chap. 2. Chapter 3 builds on this information and introduces diagnostic measurements that are of interest. Methods to compute etch rate, endpoint time, and film thickness from the interferometry signal are discussed. Chapter 4 examines potential problems with, and solutions to, the previously mentioned analysis approaches which are introduced by non-ideal device conditions. Chapter 5 discusses the implemented system hardware and software. Finally, Chapter 6 offers conclusions and suggestions for future work.

# Chapter 2

# Interferometry

## 2.1 Basic Model

The interferometry signal arises from the interference of two or more beams of light reflecting off of a thin film and a substrate. Figure 2-1 shows a cross section of light reflections for thin films. Light of wavelength $\lambda$ propagates through an incident medium with index of refraction $n_0$ until it strikes a film of index $n_1$. At this point, part of the light is reflected, and the rest is transmitted and refracted according to Snell's law. This remaining light then travels through medium 1 until is hits film 2. Again, part of the light is reflected and part is transmitted and refracted. This process continues for the entire structure. The observed light reflected off of the structure is the complex sum of all of the reflected beams.

The two beams of light have different phases due to the extra distance that one beam has traveled relative to the other. This distance is commonly referred to as the optical path difference. The optical path difference is dependent on the incident angle $\theta_0$, the indices of refraction $n_0$ and $n_1$, and the film thickness $d$, and is equal to $2n_1 d\cos(\theta_1)$. Note that $\sin(\theta_1) = \frac{n_0}{n_1}\sin(\theta_0)$ according to Snell's Law.

The observed interferometry signal is a function of film thickness, and therefore time, during an etching process. The signal is a maximum when the optical path difference is an integral multiple of the wavelength of the light. Assuming normal incidence ($\theta_0 = 0$), the maximums occur when the film thickness is an integral multiple of $\lambda/2n_1$. This number is more formally known as the period, $\Delta d$.

The ratio, $r$ (not to be confused with the Fresnel coefficients, $r_i$), of reflected amplitude to incident amplitude versus film thickness can be expressed as a relatively simple analytic

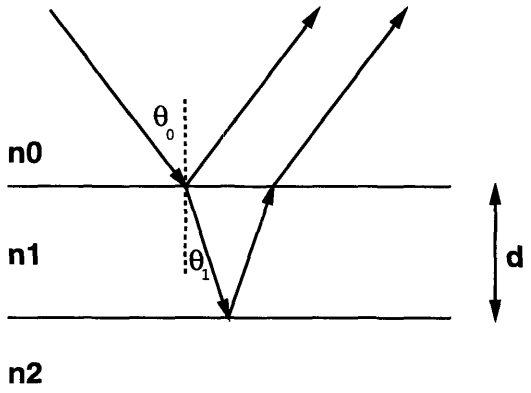


Figure 2-1: Light propagation through a thin film.

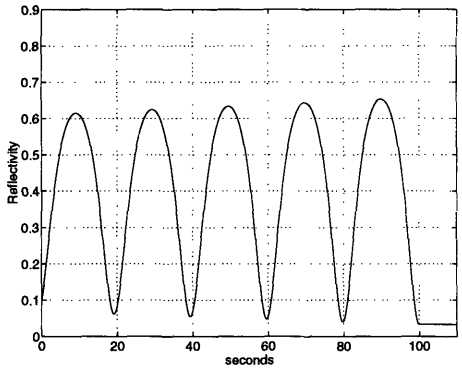


Figure 2-2: Simulated reflectivity versus time for the etching of a thin film.

expression [3, 4]:

$$r = \frac{r_1 + r_2 e^{-2i\delta}}{1 + r_1 r_2 e^{-2i\delta}} \tag{2.1}$$

where $\delta = 2\pi n_1 d/\lambda$ and $r_1$ and $r_2$ are Fresnel coefficients:

$$r_i = \frac{n_{i-1} - n_i}{n_{i-1} + n_i}.$$

The observed interferometry signal is an intensity, which is the amplitude squared:

$$R = r^2 = \frac{r_1^2 + r_2^2 + 2r_1 r_2 \cos(2\delta)}{1 + r_1^2 r_2^2 + 2r_1 r_2 \cos(2\delta)}. \tag{2.2}$$

This equation for $R$ models the interferometry signal for a three layer structure as in Fig. 2-1. An example signal for this three layer model is shown in Fig. 2-2. This model uses a wavelength of 7534Å to monitor a structure with a vacuum layer ($n_0 = 1$), on top of 5000Å of polysilicon ($n_1 = 3.73 - 0.02i$), on top of silicon dioxide ($n_2 = 1.45$), and assumes an etch rate of 3000Å/minute. Polysilicon on an oxide substrate is not a typical microelectronic structure, but is good for illustration of thin film reflections.

Given that 5000Å of poly are etched at 3000Å/minute, the endpoint occurs at 100 seconds. After the endpoint, the structure consists of only vacuum on top of oxide, thus there is only one reflected beam and no interferences are seen. Endpoint is discussed in detail in the next chapter, however, there is one topic which should be illustrated now. Note that the endpoint occurs when the signal is at a minimum. It has been proven using simple calculus that the endpoint always occurs at a minimum when the substrate's index

of refraction is less than that of the thin film. Likewise, the endpoint occurs at a maximum when the substrate's index of refraction is greater than that of the thin film, for example if oxide was being etched on a silicon substrate.

Regardless of whether the endpoint occurs at a maximum or minimum, knowing that it occurs at a zero slope point makes endpoint detection very simple. Unfortunately, more accurate modeling shows that endpoint does not necessarily occur at such a location, thus making detection more difficult.

## 2.2 Multilayer Model

Our research concentrates on the plasma etching of polysilicon over a thin oxide, thus requiring a four layer model: vacuum, poly, oxide, silicon (Fig. 2-3). The modeling of this four layer structure becomes too complex to write as a single analytic expression, and becomes even more so as more film layers are added to the model. An infinitely extensible method by Heavens [4] provides the necessary algorithm to simulate the reflectance for this four layer situation and also for any arbitrary one dimensional structure. This algorithm relies on an iterative approach to determine the reflections and transmissions at each film interface. Appendix A contains the C language implementation of the matrix model.

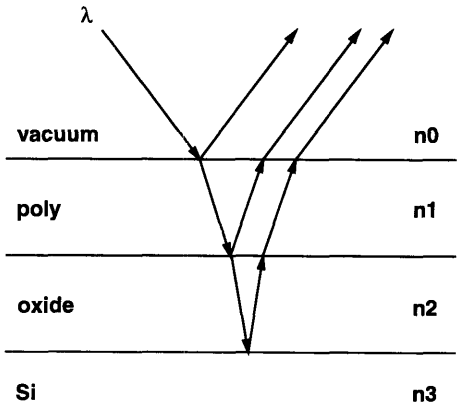


Figure 2-3: Light propagation through a cross-section of a wafer.

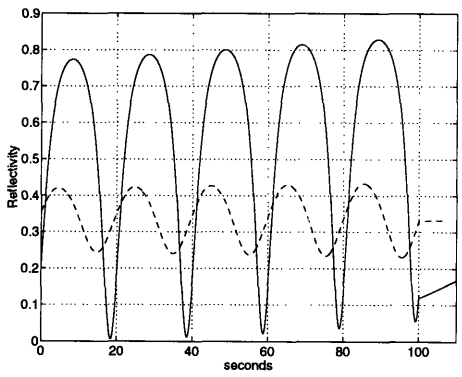


Figure 2-4: Simulated interferometry signal for the etching of 5000Å of poly on top of 1000Å (solid) or 100Å (dashed) of oxide on a silicon substrate.

The simulated interferometry signal for 5000Å of polysilicon on top of 1000Å (solid) or 100Å (dashed) of silicon dioxide is shown in Fig. 2-4. This simulation assumes an etch rate of 3000Å/min, resulting in polysilicon reaching endpoint at exactly 100 seconds. Due

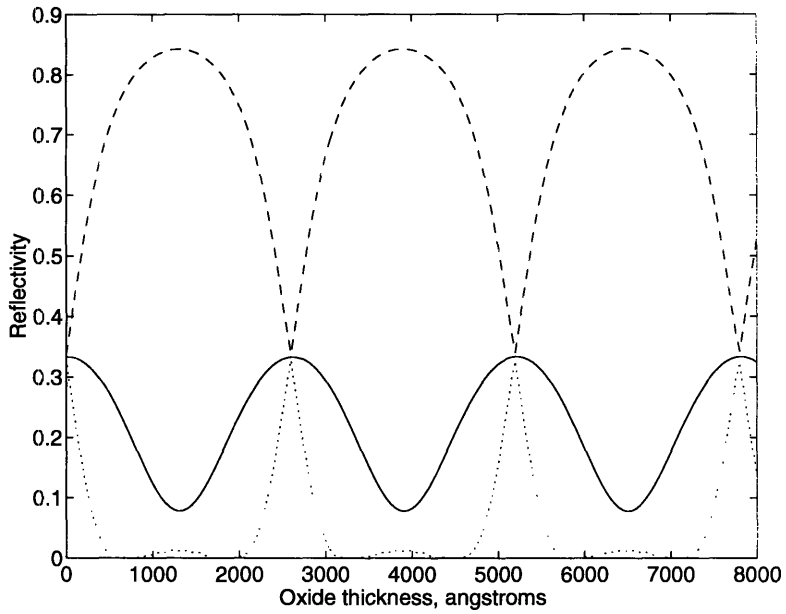


Figure 2-5: Simulated interferometry signal intensity at endpoint (solid) of polysilicon etch, along with the maximum (dashed) and minimum (dotted) intensities, as the oxide thickness is varied, assuming a monitoring wavelength of 7534Å.

to the complexity of multiple films, the endpoint does not necessarily occur at a minimum point as it did using a three layer model. Instead, the endpoint occurs somewhere between the minimum point and the midpoint of the signal if the index of refraction of the film being etched is greater than that of the film beneath, as seen in Fig. 2-5. The solid line shows the simulated signal's intensity at endpoint as the underlying oxide thickness is varied. The dashed line shows the signal's maximum intensity during the etch for each oxide thickness, and the dotted line shows the signal's minimum intensity. This simulation assumes a wavelength of 7534Å, and 5000Å poly ($n_1 = 3.73 - 0.02i$) on oxide ($n_2 = 1.45$) on a silicon substrate ($n_3 = 3.73 - 0.01i$). The period of this signal is $\lambda/2n_1$.

## 2.3 Features of the Model

Modeling is dependent on an accurate index of refraction for each film layer. This research uses values of 3.73-0.02$i$ for polysilicon, 1.45 for silicon dioxide, and 3.73-0.01$i$ for silicon [5]. These numbers are based on a wavelength of 7534Å, but can vary from wafer to wafer as much as perhaps 10% due to processing variations.

### 2.3.1 Index of Refraction Dependence

Figure 2-6 shows the effects of a 5% increase and decrease in the polysilicon index of refraction. For a change in the polysilicon index of refraction, the phase and period of the interferometry signal during the etch are affected, while the endpoint characteristic remains the same. Without accurate knowledge of the index of refraction, an incorrect etch rate would be computed. (See Sect. 3.1 for a detailed discussion on etch rate.) However, a change in the index of refraction of silicon dioxide (Fig. 2-7) only affects the amplitude of the signal, with the frequency remaining the same. The phase of the signal at endpoint is also affected, which would possibly impact an endpoint detection algorithm, depending on the algorithm, as discussed in Sect. 3.2. Etch rate calculation results do not change, as discussed in Sect. 3.1.

The effects of the imaginary part of the index of refraction (the extinction coefficient) can be seen in the interferometry signals (e.g. Fig. 2-2) as a slight increase in the overall amplitude of the signal over time. As the film thins, it absorbs less light leading to an increase in reflected intensity.

### 2.3.2 Wavelength Dependence

The interferometry signal also changes when a different wavelength of light is examined. The period of the signal is dependent on the wavelength being monitored and the real part of the index of refraction; the amplitude of the signal is dependent on the imaginary part (Fig. 2-8). Also, the published indices of refraction are dependent on wavelength. For example, the polysilicon index is $3.73\text{-}0.02i$ at 7534Å, and is $4.42\text{-}0.2i$ at 4800Å [5]. In the original case (monitoring 7534Å), the period is 1009Å. In other words, 1009Å of poly are etched during each cycle of the interferometry signal. There are approximately 5 cycles observed as we are etching approximately 5000Å of poly. However, using a wavelength of 4800Å (easily achieved by using a different bandpass filter), the period is 542Å. Now we see 9 full cycles. Also note the dramatic impact the increased extinction coefficient has on the amplitude.

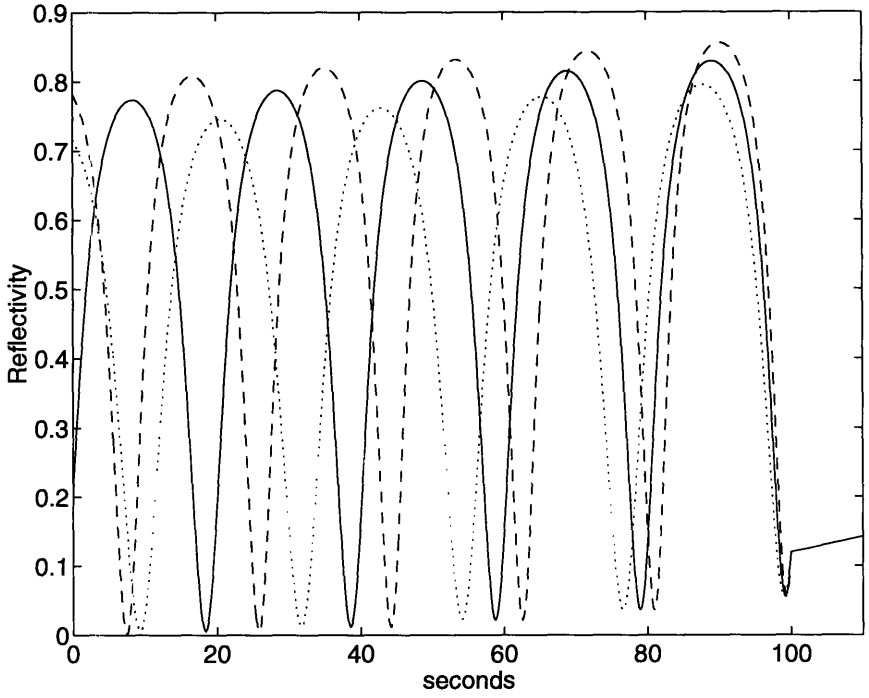


Figure 2-6: 5000Å poly (n=3.73 solid, 3.36 dashed, 4.1 dotted) on 1000Å oxide (n=1.45).

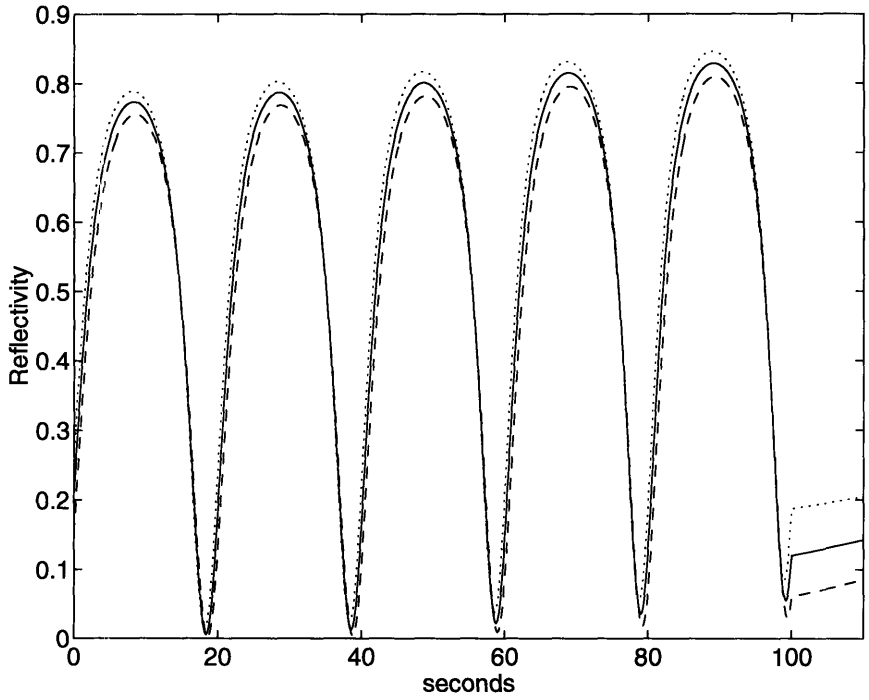


Figure 2-7: 5000Å poly (n=3.73) on 1000Å oxide (n=1.45 solid, 1.31 dashed, 1.59 dotted).

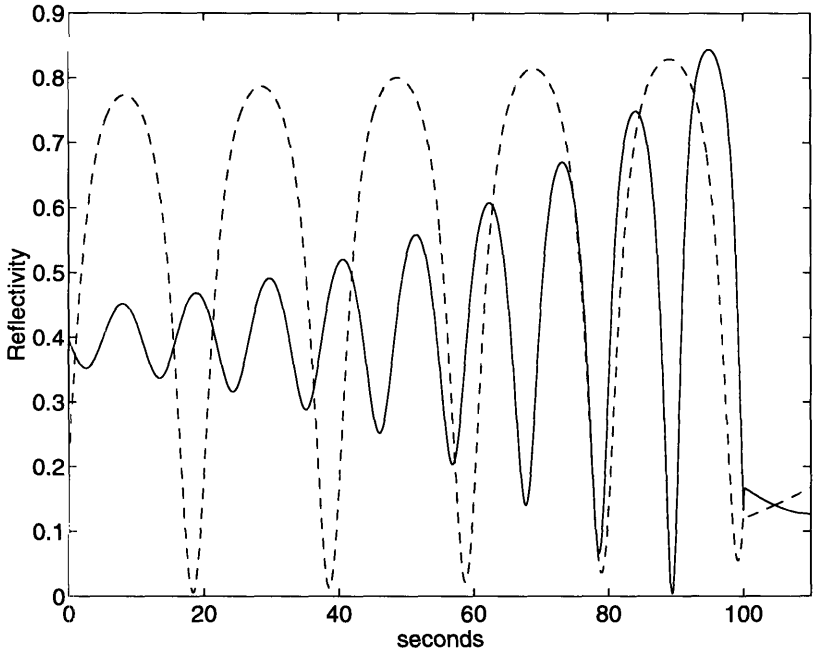


Figure 2-8: Simulated interferometry signals for 5000Å poly on 1000Å oxide monitored with 7534Å light (dashed) and 4800Å light (solid).

### 2.3.3 Underlying Film Thickness Dependence

It is interesting to note that when the thickness of a sub-layer film (*i.e.* any layer except the top or bottom) is an integral multiple of the wavelength, it has the same effect as if it had zero thickness, as if it did not exist [6]. For example, if the oxide layer in a polysilicon/oxide/silicon structure is one wavelength thick, then the structure looks like polysilicon/silicon. Since polysilicon and silicon have nearly identical indices of refraction, there is virtually no interferometry signal. Such a structure must be avoided if interferometric diagnostics or controls are desired for processing.

An extension to this is the periodicity of the amplitude of the interferometry signal with respect to the underlying oxide thickness. Figure 2-9 (which is identical to the dashed line in Fig. 2-5) shows the periodicity of the peak amplitude of the interferometry signal as the oxide thickness is varied. Clearly, oxide thicknesses near multiples of 2500Å should be avoided when using a monitoring wavelength of 7534Å (as the maximum and minimum intensities achieved during a process are nearly the same). Likewise, oxide thicknesses near odd multiples of 1250Å are optimal for monitoring purposes.

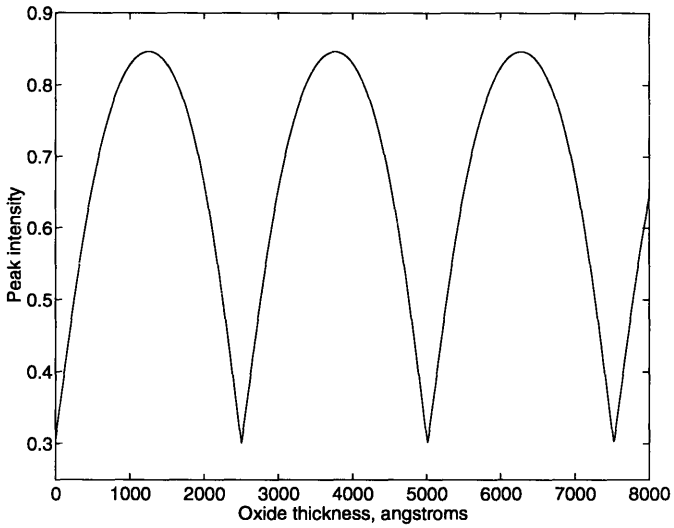


Figure 2-9: Periodicity of the peak value of the interferometry signal as the underlying oxide thickness is varied.

## 2.4 CCD Interferometry

Laser interferometry uses laser light which is reflected by the wafer and collected by a photo-detector. Another variation is optical emission interferometry, which uses the radiation emitted by the plasma as the light source to the photo-detector, after passing through a bandpass filter. This research takes optical emission interferometry a step farther by using a CCD camera in place of the photo-detector [7]. While a laser must be aimed at a specific wafer location, the CCD allows the monitoring of any point on a wafer by choosing the desired pixel. With the exception of the light source, each pixel of the CCD interferometer functions like a laser interferometer focused on that particular point on the wafer (Fig. 2-10).

Unfortunately, there are some side effects to using the plasma emission as the light source for the camera. The plasma emission, unlike a laser beam, is not constant during the process. In fact, the emission's intensity rises significantly near endpoint. The rise in optical emission spectroscopy (OES) at endpoint is due to the change in concentration of source gas by-products (e.g. reduction in consumption of poly etchant species and increase in oxide etch by-products). As the intensity of the light source increases, the reflected intensity sensed by the CCD follows. When a pixel that reaches endpoint early relative to the rest of the wafer is examined, little distortion is seen. However, when a pixel that has a relatively late endpoint is examined, it is possible that the endpoint characteristic is

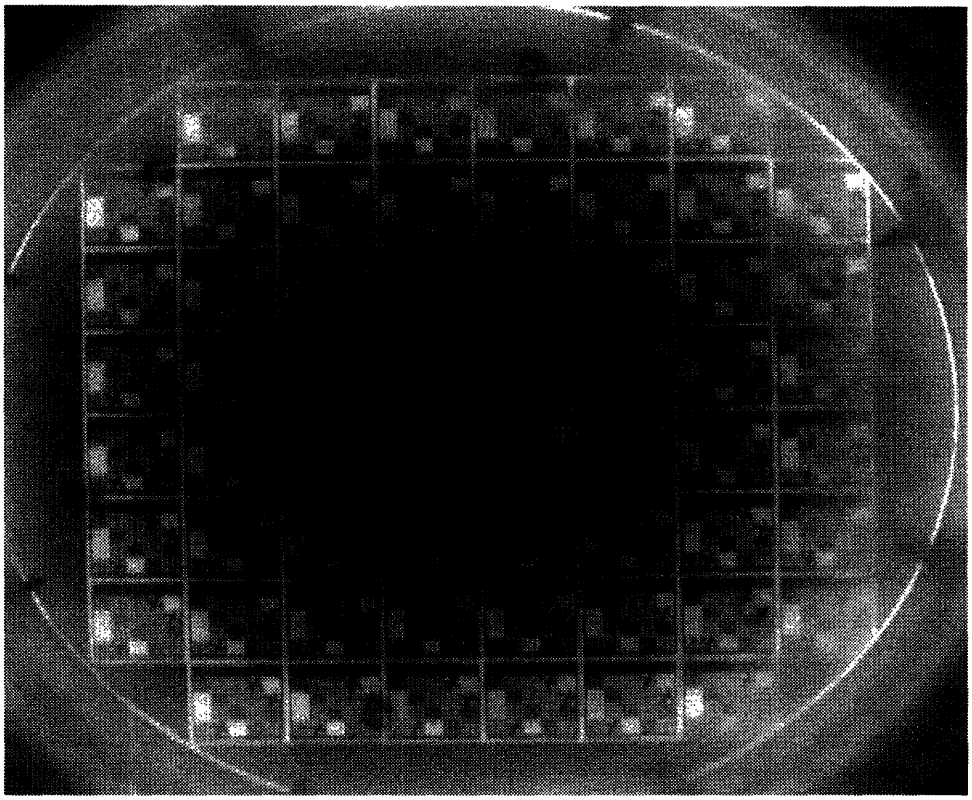

Figure 2-10: CCD image at 17 seconds into the etch.

masked by the rising plasma emission. Figure 2-11 illustrates this point by showing the sum of a simulated interferometry signal and a mock optical emission signal which rises shortly before the actual endpoint. While the sum of these two signals does not model the physical situation accurately, it is close enough and illustrates the point more easily. The original signal (a) has an endpoint at 100 seconds, while the modified signal's endpoint (b) occurs at 105 seconds. An endpoint characteristic determined by the signal going "flat" appears in the wrong place due to the effects of the plasma emission. Model-based endpoint detectors, if they account for this phenomenon, are able to correctly detect the endpoint. Research into integrating OES data and interferometry data is currently in progress at MIT.

Despite this undesired behavior of the CCD interferometry method, there are many advantages over laser interferometry that make the CCD system worth using. Briefly stated, they are:

- Setup time is faster because no laser alignment to a specific location needs to be performed.
- Uniformity measurements can be obtained by monitoring many points on the wafer.

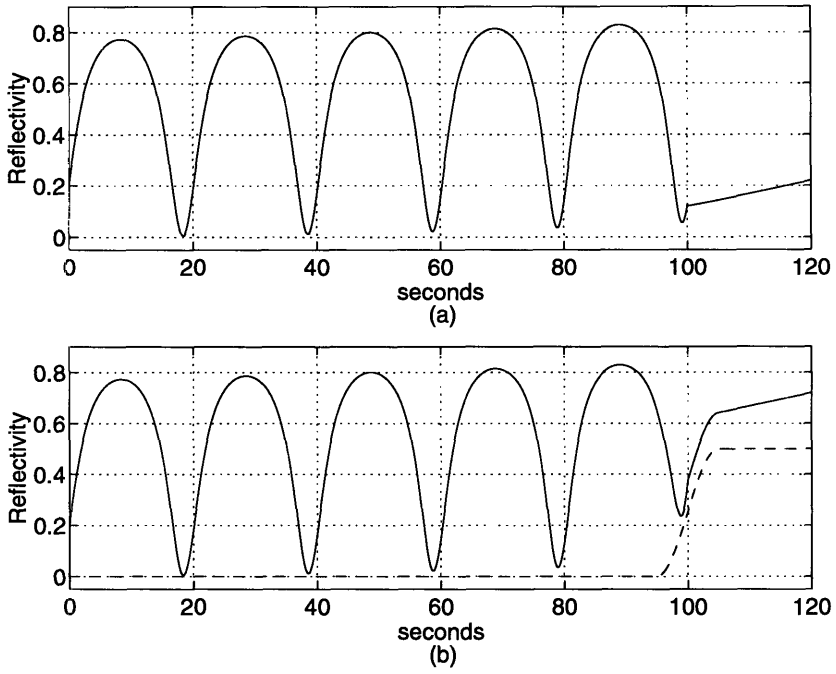


Figure 2-11: (a) A simulated interferometry signal using a constant light source. (b) The same interferometry signal (solid) affected by a rising plasma emission signal (dashed).

- With an endpoint algorithm, one could know when a given percentage of the wafer has cleared. This information could be used to know when to switch chemistries in the final stages of the etch.
- The potential for real time process control is available.

## 2.5 Conclusion

This chapter has presented the interferometry signal and discussed some of its features. Chapter 3 discusses ways of building on the feature knowledge to extract useful information, such as etch rate and endpoint, out of the interferometry signal.

# Chapter 3

# Analysis Approaches

The previous chapter introduced the interferometry signal and its characteristic features. This chapter builds on these features by discussing algorithms for extracting three useful pieces of information out of the interferometry signal: how fast the film is etching (etch rate), when the film is finished etching (endpoint), and the thickness of the film being etched.

## 3.1 Etch Rate

As discussed in the previous chapter, the period, $\Delta d$, of the reflected intensity versus film thickness is $\lambda/2n_1$. This number is the amount of material that has been etched over the period. We can determine the time period of the intensity versus time signal (the observed interferometry signal) by measuring, for example, the time between adjacent minima, $\Delta t$. The etch rate is now easily computed as $ER = \Delta d/\Delta t$.

The time period is more accurately determined by taking the Fourier transform, implemented using a Fast Fourier Transform (FFT), of the signal. After removing the constant bias (the zero frequency component) of the signal, the fundamental frequency is the frequency with the largest FFT magnitude. This is more accurate than the "time between adjacent minima" because it is immune to high frequency noise, which masks the exact location of a minima (or maxima) in the time domain. Figure 3-1 shows the magnitude of the 2048 point FFT of a sample interferometry signal.

While a Fourier transform ideally provides exact frequency information, the FFT only provides information at a fixed set of frequencies; the FFT can be thought of as a sampled

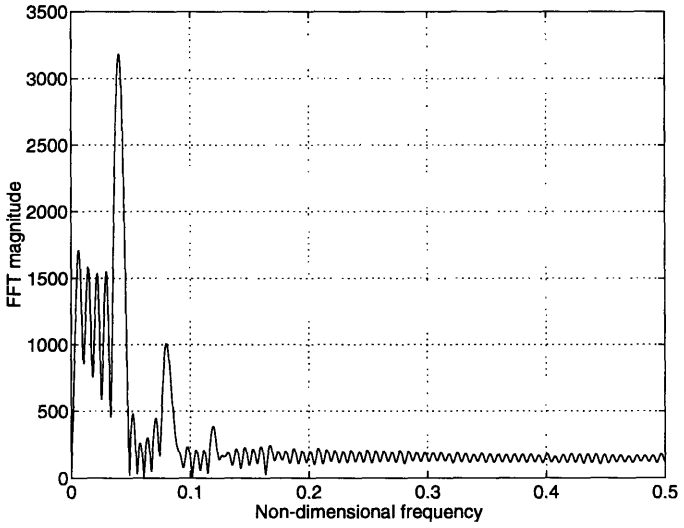


Figure 3-1: Magnitude of 2048 point FFT of interferometry signal. The fundamental frequency is 0.04.

version of the Fourier transform. This sampling introduces numerical inaccuracies to the etch rate computation since the etch rate can now only be known to within a quantized range. The larger an FFT used to calculate etch rate, the more accurate the result, at the expense of computation speed. Depending on the use (accuracy is more important for post-process diagnostics, while speed is more important for real time control calculations) a 512, 1024, or 2048 FFT provides adequate results.

The etch rate equation using the acquisition system described in Chap. 5 is

$$ER = \frac{\lambda}{2n_1} \times f \times FR \tag{3.1}$$

where $f$ is the fundamental frequency of the camera, expressed as a number between 0 and 1 (corresponding to frequencies between 0 and $2\pi$), and $FR$ is the frame rate of the camera [7]. Note that $f \times FR = 1/\Delta t$.

The etch rate is *average* in the sense that it is computed by taking the FFT over the entire signal. (Or, similarly, by determining the time between the first minima and the last minima, and dividing by the total number of minima minus one). This *average* method can not detect any etch rate variations during the process, and thus has limited potential for real time control.

### 3.1.1 Windowed Etch Rate

A logical extension to the average etch rate computation uses a moving window to determine the etch rate for only a portion of the interferometry signal, and then moves that window for the next portion. Basic signal processing theory dictates that at least one period of a signal is needed for accurate processing. Assuming a 1Hz frame rate for simplicity, and approximately 5 cycles of the interferometry signal in 100 seconds, a 20 point window could be used. A real time etch rate computation algorithm could acquire 20 points, then take the FFT and compute the etch rate. Upon getting the next data point, the window would move over by one point to compute the FFT of points 2 through 21. This process would continue for the entire length of the signal.

Unfortunately, instantaneous etch rate computation is not that easy. The resulting etch rate versus time plots show the etch rate varying periodically over the plot (Fig 3-2). In fact, when a simulated interferometry signal (created with a constant etch rate) was processed, it, too, had a varying etch rate versus time. The two reasons for this processing error are window length and window shape.

Plots of the windows in the frequency domain illustrate why window shape affects the FFT calculation (Fig 3-3). A rectangular window has side lobes which are approximately 20% of the main lobe magnitude. These side lobes may overlap due to aliasing caused by discrete time sampling. However, the Hamming window side lobes are only 1% of the main lobe magnitude, causing less signal distortion.

Having established that the Hamming window is most optimal, we must investigate the effects of FFT length and window length. As discussed earlier in this section, an FFT is a sampled version of the Fourier transform. Thus, the computed fundamental frequency is affected by quantization error. For example, assume that the 512 point FFT of an interferometry signal has its maximum frequency at point 25, corresponding to a frequency of $0.0244\pi$, or a non-dimensional frequency of 0.0488. That same interferometry signal processed with a 1024 point FFT may have its maximum frequency at point 49, 50, or 51, corresponding to fundamental frequencies of 0.0479, 0.0488, or 0.0498. Multiplying by the constants as in (3.1), these frequencies correspond to etch rates of 2903Å/min, 2957Å/min, or 3018Å/min, respectively. Quantization errors such as these indicate a need for a higher point FFT to yield a more precise fundamental frequency.

While more precision is desired for a diagnostic etch rate, the extra precision may in

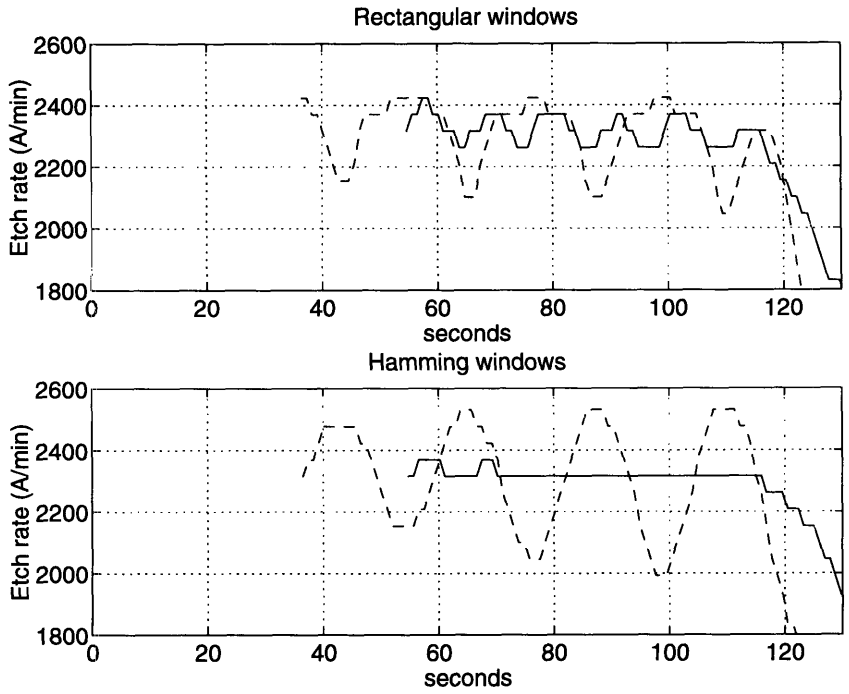


Figure 3-2: Etch rate versus time using 55 second windows (solid) and 35 second windows (dashed) for an experimental interferometry signal.

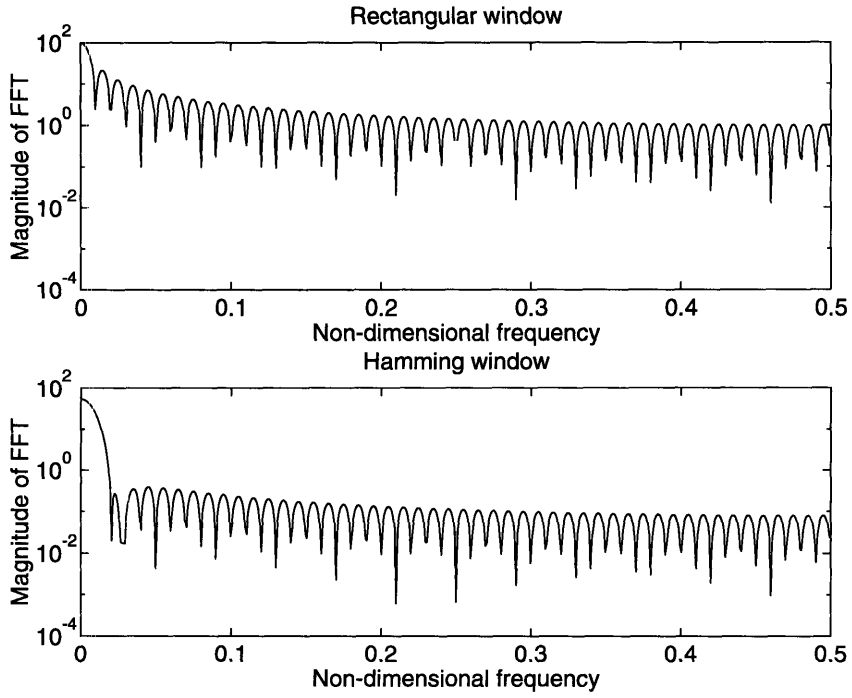


Figure 3-3: FFT magnitudes for rectangular and hamming windows.

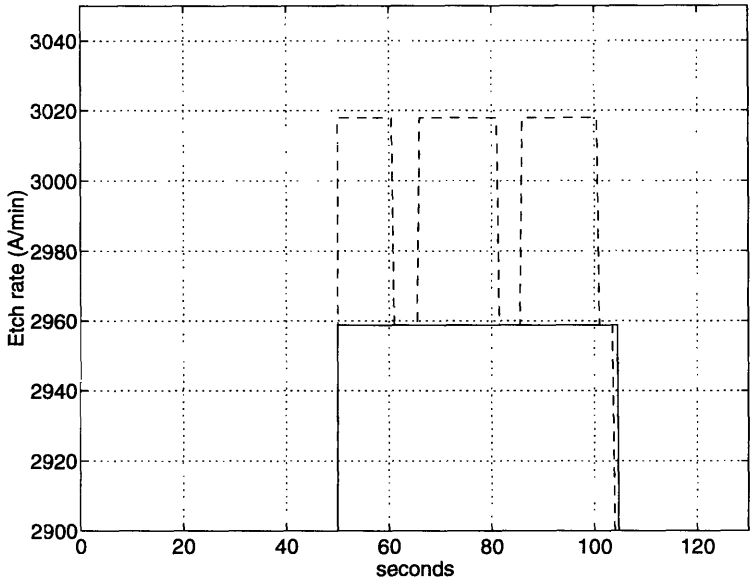


Figure 3-4: Etch rate versus time for a simulated interferometry signal. The actual etch rate is 3000Å/min, but quantization noise of the 1024 point FFT (solid) and 2048 point FFT (dashed) prevent the exact number from being computed.

fact be detrimental to a controller when used with the moving FFT window. This point is illustrated in Fig. 3-4, which shows the windowed etch rate for a simulated interferometry signal. This simulation built a signal assuming an etch rate of 3000Å/min, and used a window length of 50 seconds. Ideally, both lines should be at exactly 3000Å/min, although neither can be at exactly 3000 due to quantization levels. The result is that the 1024 point FFT is able to be constant at 2958 (its nearest possible values are 3131 or 2785), while the 2048 point FFT fluctuates between 2958 and 3018 as it tries to get to exactly 3000. The fluctuation between the two values could cause a controller to perform unnecessary process adjustments.

In principal, a smaller window provides a more instantaneous result because it is averaging over a smaller set of data. Unfortunately, a smaller window also leads to inaccuracies in computation due to numerical errors such as quantization noise. As shown in Fig. 3-5, a larger window removes these undesirable effects by averaging the signal over more time. The 30 second window (dotted) fluctuates between 2000 and 3500Å/min. The 40 second window (dashed) is a great improvement, but still fluctuates a little. The 50 second window is constant, as it should. These simulated plots were made using the same interferometry

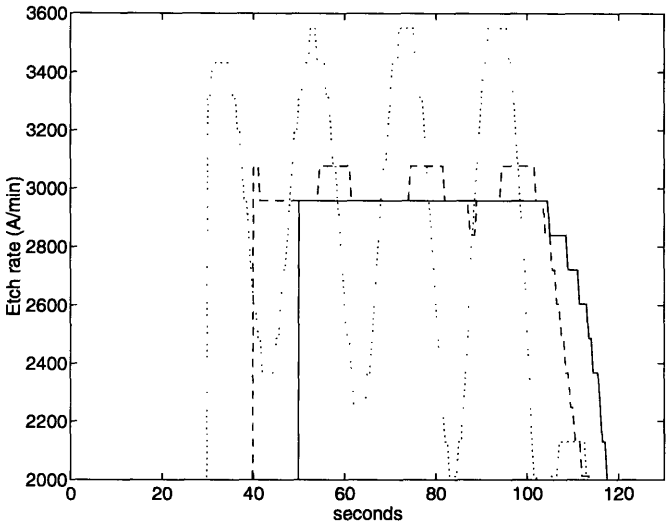


Figure 3-5: Etch rate versus time for a simulated interferometry signal using a 1024 point FFT. The actual etch rate is constant at 3000Å/min, but 30 second (dotted) and 40 second (dashed) windows are too short to yield a constant value, as with the 50 second (solid) window.

signal as the previous example; it assumed an etch rate of 3000Å/min, resulting in a period of 20 seconds. This figure concludes that approximately two and a half cycles of a signal are needed for accuracy when using the windowed FFT method. Note however that as the window length grows, it approaches the average etch rate method discussed earlier, and thus offers little, if any, potential for real time control decisions.

### 3.1.2 Frame Rate Limitations

A limitation of the implemented system is the frame rate of the CCD camera. Only a fixed number of images can be acquired in a given time, thus constraining the sampling rate of the interferometry signal (pixel intensity versus time). Theoretically, the etch rate calculation can account for any frame rate as long as the frame rate is greater than the Nyquist frequency of the interferometry signal. However, model fitting is much more accurate with a higher frame rate. In addition, note that as the frame rate is increased, a proportionally higher point FFT is required to achieve the same quantization levels, as evident in (3.1).

There are several limiting factors on the frame rate. The first, as discussed in the preceding paragraph, is a bandwidth constraint. The second is a process constraint. The AME-5000 uses a rotating magnetic field to achieve azimuthal uniformity; that is, the etch

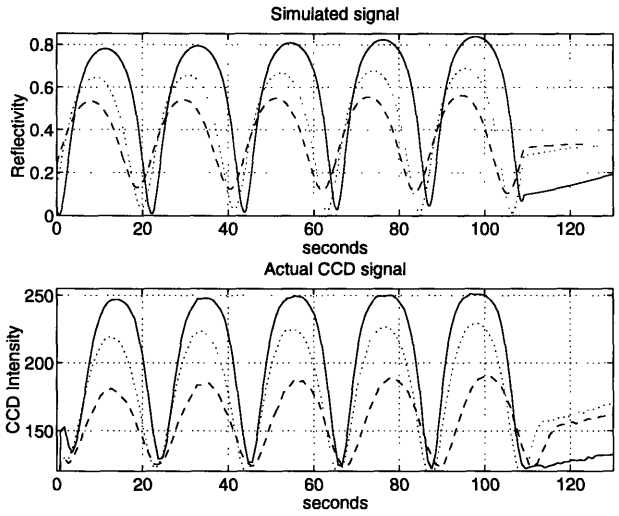


Figure 3-6: Simulated and experimental interferometry signals for etching 5100Å poly on 1100Å (solid), 450Å (dotted), and 250Å (dashed) oxide.

rate is the same along circumferential lines on the wafer. The plasma intensity in the etching chamber is brighter near the vicinity of the magnetic field's poles, and this bright spot revolves around the chamber as the magnetic field rotates. In order to eliminate the extra source of brightness variation, the CCD camera is triggered to take a picture each time the magnetic field passes the same location in the chamber. Thus, the frame rate is limited by the rate of the magnetic field. The magnetic field currently rotates at approximately 2Hz, but there are plans to increase this two- or four-fold in the near future.

## 3.2 Endpoint Detection

For plasma processing, OES is a commonly used endpoint detector. It provides average information about the entire wafer. Using the CCD camera, endpoint can be determined at any location by observing a significant change from the previous interferometry points. Figure 3-6 shows simulated and actual interferometry signals. 5100Å of polysilicon are etched at a rate of approximately 2800Å/min, resulting in an endpoint near 110 seconds. The signal's characteristic change at endpoint (where the signal becomes "flat") results from the different film properties (e.g. the refractive index) along with the slower etch rate observed when etching the oxide compared with poly.

### 3.2.1 Algorithms

While the signal change at endpoint is usually obvious to a human viewer looking at the entire signal, it is not so simple for a computer to "see" this change when it occurs without having knowledge of the rest of the signal. The challenge is to minimize the amount of time after endpoint occurs needed by a computer (or human) to recognize the change.

Three algorithms for endpoint detection are investigated: windowed FFT, hypothesis testing, and local min/max detector.

#### Windowed FFT.

The simplest endpoint detector is frequency based; it looks for significant changes in the frequency of the signal (which correspond to changes in etch rate). This method works because the oxide etches at a slower rate than the poly. Using a windowed FFT as discussed in Sect. 3.1.1, the fundamental frequency is monitored during the process. Endpoint is indicated when the fundamental frequency falls considerably.

However, this method has some problems. First is the windowing problem previously discussed. Second is that at least half of the window must be over the oxide region of the interferometry signal before the oxide's slower etch rate can be detected. Given the relatively large window size, this algorithm is not be able to indicate the endpoint until at least 15 seconds after it has occurred. This delay is acceptable for a post-process diagnostic which need only identify the endpoint time. However, the delay is not satisfactory for a real time endpoint detection system which controls the process, and presumably turns the machine "off" upon sensing the endpoint.

#### Hypothesis Testing.

Another possible algorithm for endpoint detection is a statistical hypothesis test. This algorithm uses a simulated model for etching poly, which is compared to the actual interferometry signal (Fig. 3-7). Knowledge of the film thicknesses and indices of refraction are necessary to build the model. The algorithm waits for a full period of the interferometry signal, and then builds a model on the fly based on the experimentally computed etch rate. The model signal always etches poly, and thus has a constant etch rate throughout the process which is exactly the etch rate of the actual signal. However, the actual signal changes

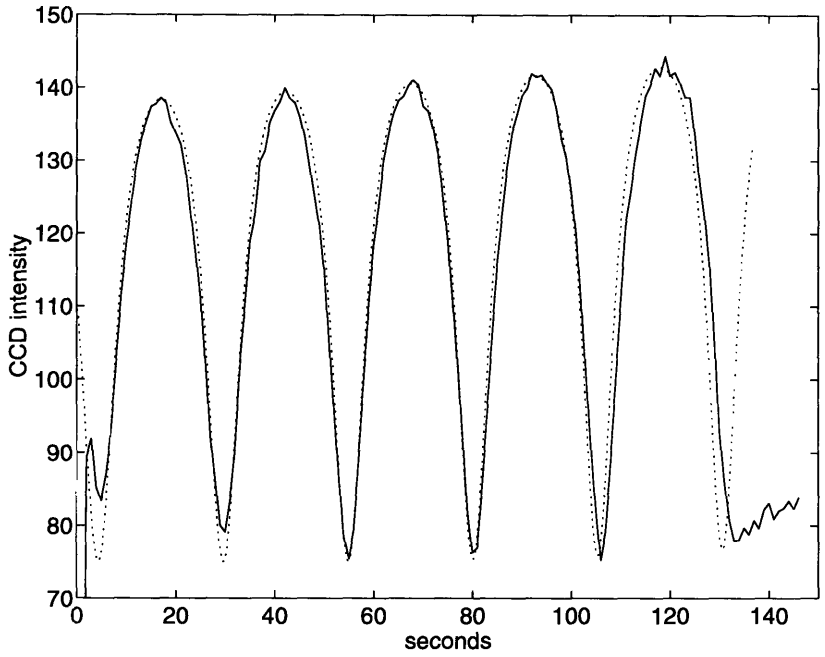


Figure 3-7: The hypothesis test compares an adaptive model signal (dotted) with the experimental signal (solid). The two signals are "close" for most of the run, but differ significantly at endpoint (130 seconds).

frequency upon reaching endpoint, and the two signals are no longer the same (or "close to" the same). The hypothesis testing method is able to determine what is "close" and apply this decision to endpoint detection. The hypothesis test ignores most noise, but there is a potential for large noise to be interpretted incorrectly as a signal change.

### Local Min/Max Detector.

The third detection algorithm also uses pre-determined modeling knowledge. Unlike the hypothesis test, which makes use of a constant comparison between the model and the actual data, this algorithm uses a model which describes how many cycles appear until endpoint, and the approximate distance from the last local minimum to the endpoint location. The full model signal is no longer needed after this information is obtained. Then, as the actual data comes in, the algorithm detects and counts local minima. When it "knows" that the last minimum has occurred, it can make an approximate guess of the remaining time until endpoint.

This algorithm is very sensitive to noise (which may lead to additional local maxima),

so a smoothing algorithm should be applied to the signal. However, a smoothing algorithm (either low pass filtering or a 3 point moving average filter in the time domain) requires future knowledge of the signal. This requirement adds a delay (though not nearly as large a delay as with the windowed FFT based endpoint detector) to the endpoint detector.

Another disadvantage is that this method is more dependent on precise modeling than the others. Simple modeling can accurately predict the number of minimums that will occur during a given run, but the exact phase of the signal at endpoint (i.e. the exact amount of time between the endpoint and the last minimum) requires very precise indices of refraction, as illustrated in Fig. 2-7. It also assumes that the etch rate is constant and the same as the etch rate from the rest of the run. This assumption is not good because a slower, more precise etch may be used as the endpoint nears.

An extension to the min/max method detects the change in etch rate via derivatives. The slower etch rate of the oxide appears close to "flat" in contrast to the steep slope of the signal between the last minimum and the endpoint. This algorithm has some problems due to a discretized derivative computation, and is also very sensitive to noise. However, the combination of the local minimum detector for most of the process, and the "flat" sensor (via derivatives) for the last few seconds, provides the most robust endpoint detection algorithm because it is the most flexible and least dependent on precise indices of refraction.

**Full Wafer Endpoint.**

The above algorithms detect endpoint at a single point on the wafer. These algorithms are easily extended to a full wafer endpoint detector by simultaneously monitoring several strategically placed points on the wafer. Ha [2] measured the etching rate at 12 points across a wafer as shown in Fig. 3-8. Using these 12 points, placed along three concentric rings on the wafer, he defined the slope of the radial uniformity to be the difference between the average etch rates of the four points on the outer ring and the average etch rates of the four points on the inner ring, all divided by the average etch rate of all twelve points. A slope of zero means that the wafer is etching uniformly, assuming a concave or convex etch across the wafer. A positive slope indicates that the outer ring of the wafer is etching faster than the inner ring, and vice-versa for a negative slope. A similar strategy can be applied to measuring endpoint uniformity.

Patel, et. al. [8] used an infrared (IR) sensitive CCD camera to measure temperature

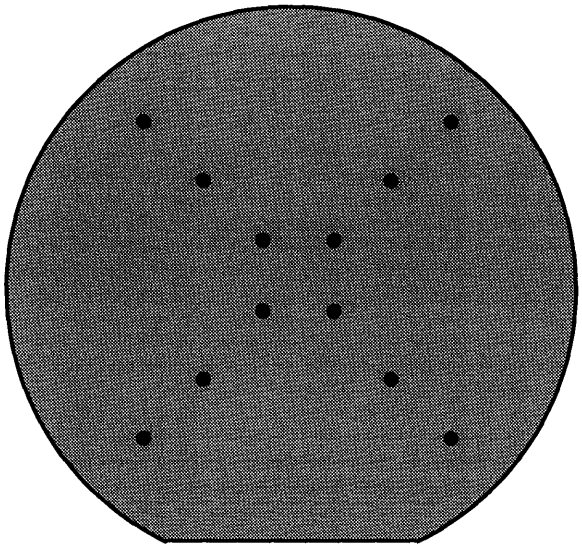

Figure 3-8: 12 points arranged in 3 radial loops around the wafer for analysis.

across a wafer. Similar to this research, they were able to save full wafer images or focus in on a single pixel's intensity over time.

Nagy [9] performed *ex-situ* experiments to determine etch rate across a wafer. Wafers were etched for a fixed amount of time, and then film thickness measurements were taken at nine equally spaced points across a diameter of the wafer. Etch rate was computed as the etch depth (initial thickness minus final thickness) divided by the etch time.

Another full wafer monitoring technique, performed by Grimard, et. al. [10, 11], used scalar diffraction to produce pseudo-Fourier transforms of the wafer surface. The transforms were then analyzed to detect changes in surface topography.

### 3.2.2 Alternative Graphing Methods

Maynard [12] discussed two novel graphing techniques, Lissajous graphs and phase-space plots, which may make the endpoint signal more easily visible to technicians. Lissajous graphs plot the signal $R(t)$ relative to $R(t-\Delta t)$, where $\Delta t$ is an arbitrary value between 0 and the period of the signal; time is implicit. Simulated (assuming 5000Å poly on 1100Å oxide on a silicon substrate, monitored with light at 7534Å) and experimental interferometry plots are shown in Figs. 3-9 and 3-10, respectively. The Lissajous graphs for these interferometry signals are shown in Figs. 3-11 through 3-14. The endpoint in these plots can be seen as

the point where the line "leaves" the rest of the plot in the lower left corner of each. These figures show that the arbitrary choice of $\Delta t$ is crucial to recognizing the endpoint. Figure 3-14 ($\Delta t = 4$ seconds) shows the endpoint very clearly, while it can barely be recognized in Fig. 3-12 ($\Delta t = 1.5$ seconds). Furthermore, the "best" value for $\Delta t$ in this signal may not be the best for a different signal. Note however that the endpoint is always visible in the simulated Lissajous graphs (Figs. 3-11 and 3-13).

Phase-space plots show the time derivative of the signal, $\frac{dR}{dt}$, relative to $R(t)$. Figure 3-15 shows the phase-space plot for a simulated interferometry signal. As in the Lissajous graph, time is implicit in the phase-space plot. Y-axis zero crossings on the left side of the plot correspond to local minima, while those on the right indicate local maxima. At zero time, the phase-space signal is in the lower left hand corner of the curve, close to -0.05 on the y-axis, and 0 on the x-axis. As time continues, the signal spirals inward due to the amplitude increase seen in the traditional interferometry plot. Upon reaching endpoint, the signal makes a sharp jump and settles near zero on the y-axis. This is because the interferometry signal is now approximately "flat," and thus has a derivative near zero. Note that it is beneficial that the slope is not exactly flat, for if it were, the phase-space plot would stand still. In other words, if the derivative of the signal is exactly zero, then the signal's intensity would be constant, and the phase-space representation would be a stationary point. A phase-space plot of an experimental interferometry signal is shown in Fig. 3-16. The jagged character of this plot is due to signal noise. We do not see a clear endpoint of this signal because it is hidden due to the zero-slope problem discussed in the previous paragraph. Note that while the phase-space plots are visually attractive to a technician, they offer no additional information to a computer algorithm that could not otherwise be obtained from a derivative.

## 3.3 Film Thickness

So far, only etch rate and endpoint analysis have been approached. Ideally, a film thickness is desired. Endpoint, or any point for that matter, is immediately known if a film thickness measurement is available in real time. Likewise, the etch rate can be trivially computed given the film thickness over time.

A number of techniques already exist to measure film thickness. Traditionally, these

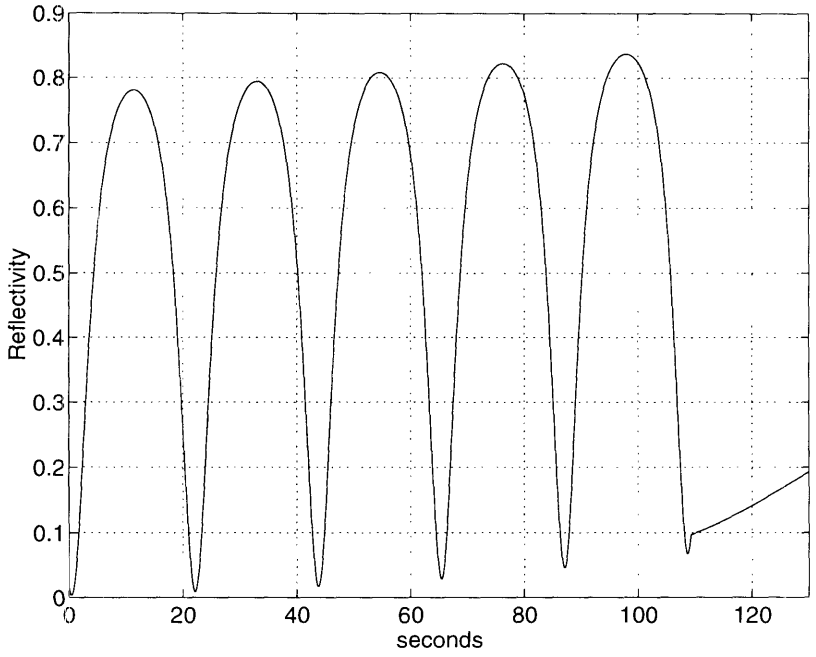


Figure 3-9: Simulated interferometry plot.

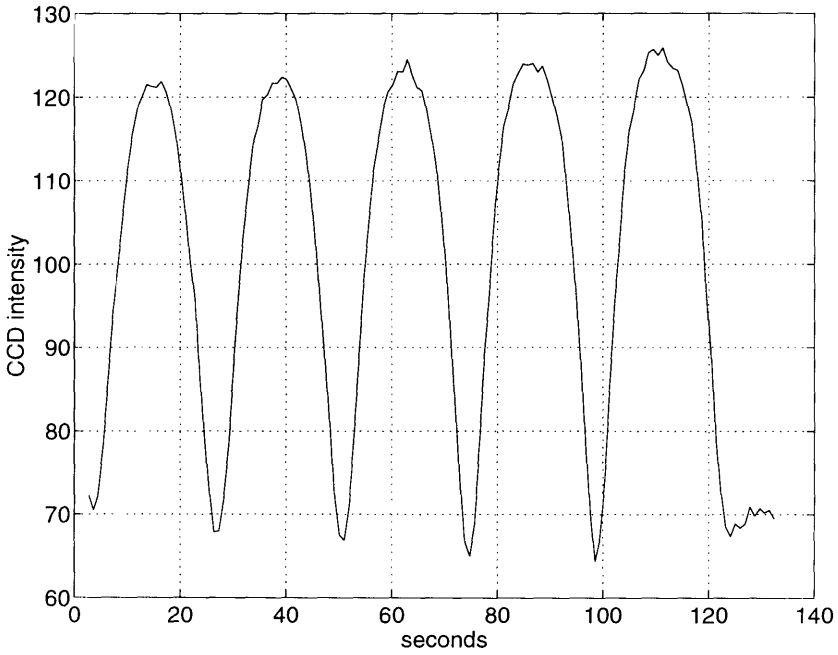


Figure 3-10: Experimental interferometry plot.

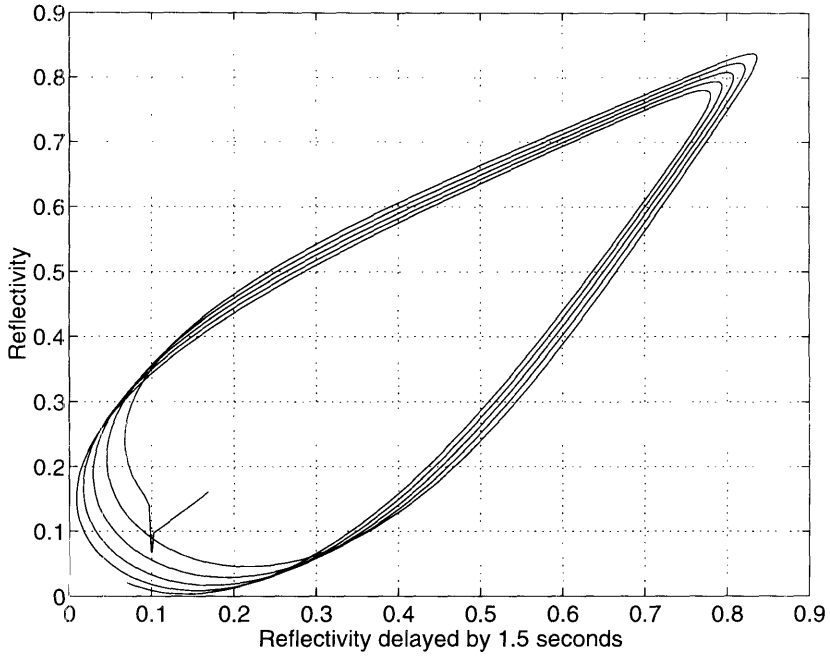


Figure 3-11: Simulated Lissajous graph ($\Delta t = 1.5$ seconds).

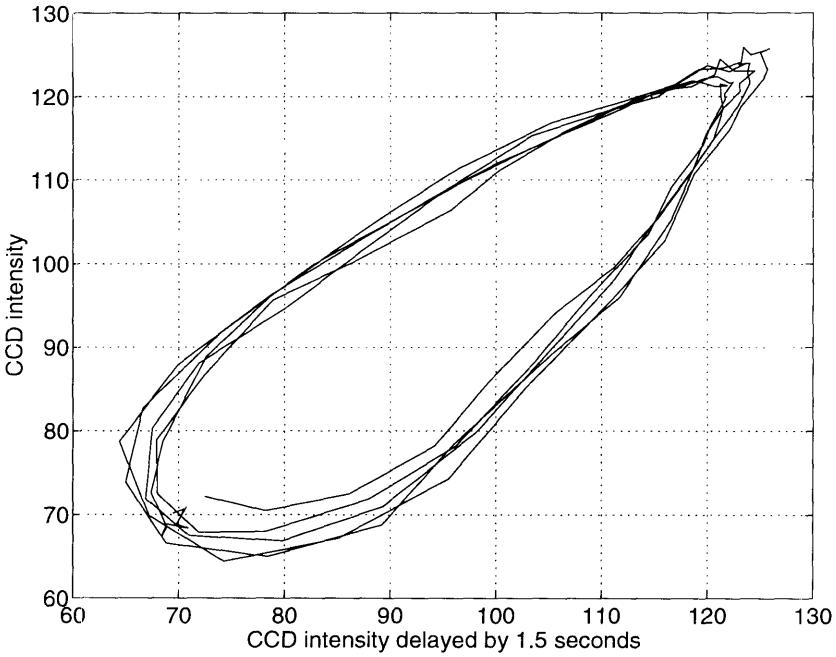


Figure 3-12: Experimental Lissajous graph ($\Delta t = 1.5$ seconds).

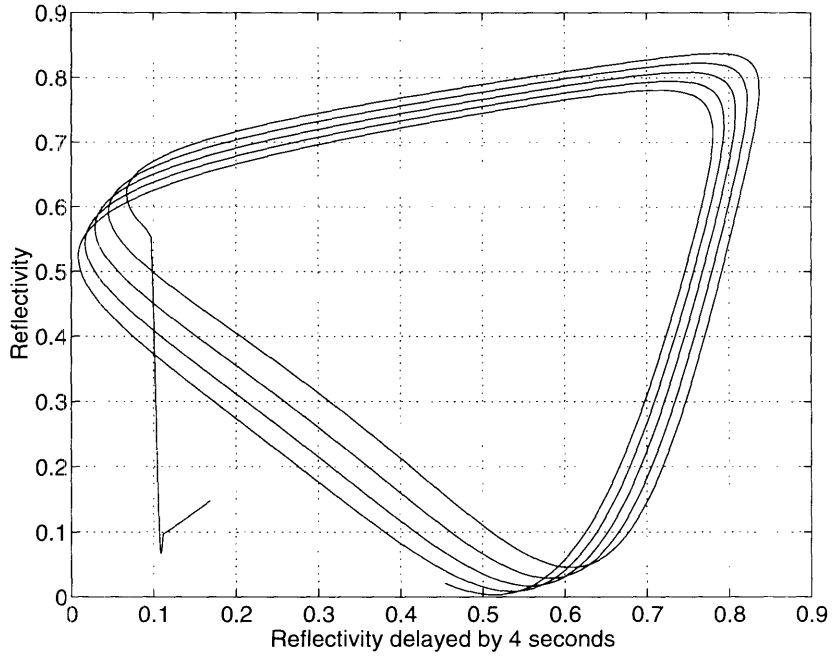


Figure 3-13: Simulated Lissajous graph ($\Delta t = 4$ seconds).

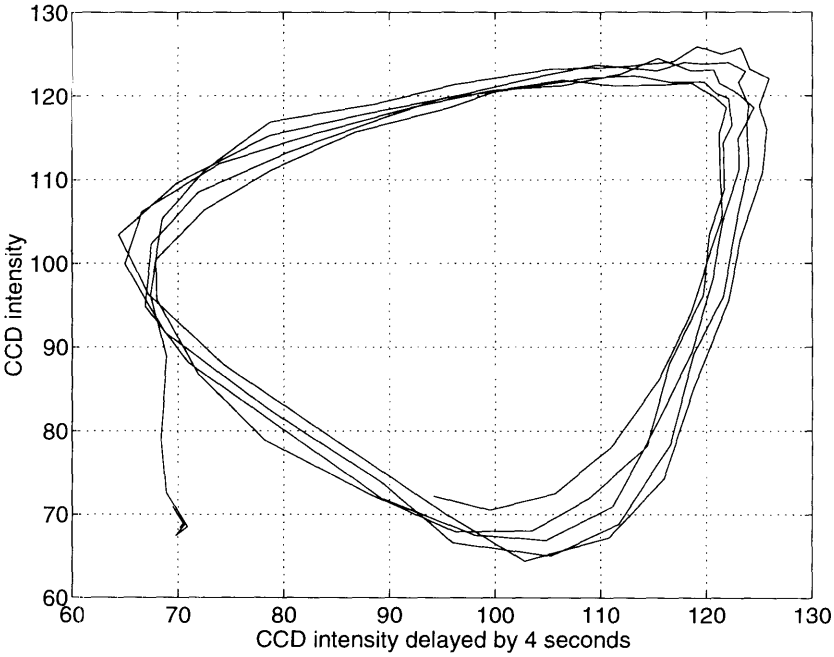


Figure 3-14: Experimental Lissajous graph ($\Delta t = 4$ seconds).

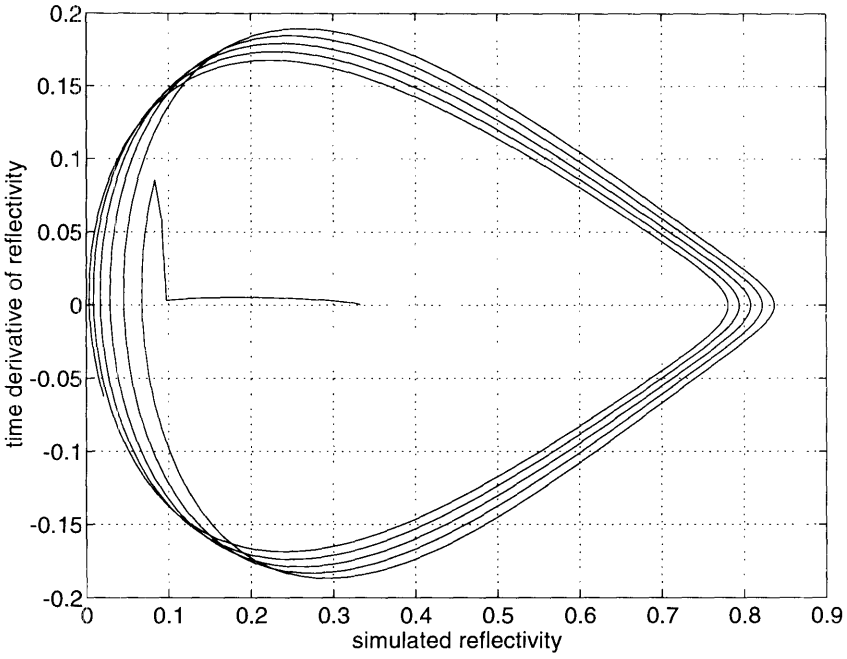


Figure 3-15: Phase-space plot for simulated signal.

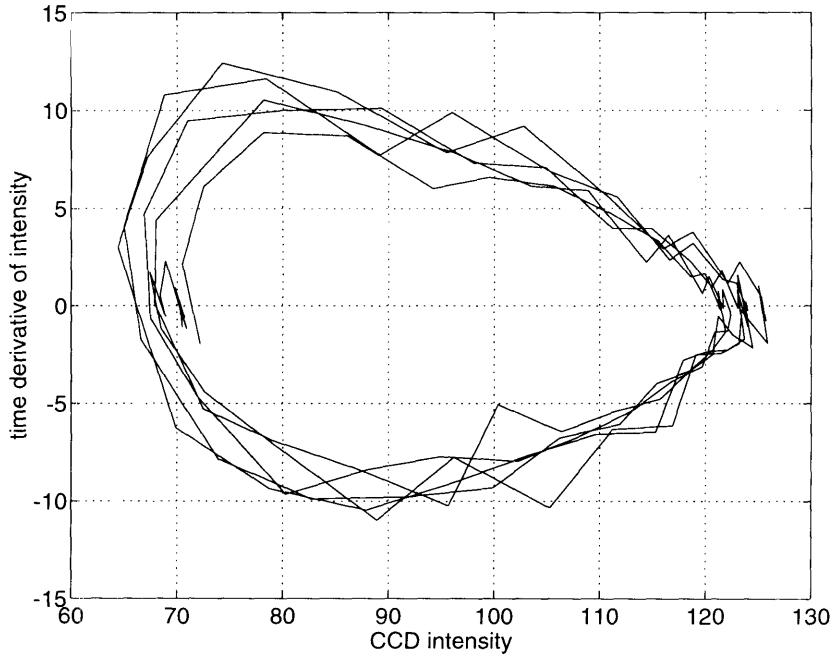


Figure 3-16: Phase-space plot for experimental signal.

methods required the use of tools such as a Nanospec or a Michelson Interferometer [13] between processing steps. Henck [14] used ellipsometry for *in-situ* monitoring for polysilicon over silicon dioxide films. Other techniques include multiple wavelengths pyrometric interferometry [15] and emission and reflection Fourier transform infrared spectroscopy (EFTIR) [16].

The algorithm for film thickness measurements using the CCD interferometer is relatively straightforward, although there are some easily overlooked pitfalls. The general idea of the algorithm is that we have an equation giving reflected intensity as a function of film thickness and wavelength. Reversing the equation, for any reflectivity, a set of possible film thicknesses can be obtained. Using pre-process information, a guess can be made to select which of the film thicknesses is correct. The implemented algorithm is as follows:

1. Acquire a period of data to determine the minimum and maximum points on the interferometry signal along with the etch rate.
2. Use known parameters (initial film thicknesses, indices of refraction, and computed etch rate) to build a simple interferometry model.
3. Using this model, attach a film thickness to each point on the model signal.
4. Correlate the experimental signal with the model signal to determine the film thickness at each point of the experimental signal.

This algorithm is affected by even the smallest deviations from the model (such as index of refraction or inaccurate initial film thickness) in addition to noise, rendering it nearly useless. An improvement can be obtained by collecting data at several different wavelengths and averaging the results. A tunable wavelength filter provides the ability for the CCD camera to monitor a series of different wavelengths during the process.

## 3.4 Conclusion

This chapter presented techniques for extracting diagnostic information from the interferometry signal. This information can ultimately be used as the input to a process control system. The techniques presented here, however, assumed mostly ideal conditions during data acquisition. Chapter 4 discusses a number of situations which require more accurate modeling to account for non-ideal conditions to achieve satisfactory analysis results.

# Chapter 4

# Modeling Issues and Experiments

Interferometric etch rate and endpoint computation rely heavily upon accurate modeling. Unfortunately, there are many situations in which a relatively simple model is not correct. This chapter describes experiments and models to identify and account for important interferometry signal effects. Wafers were fabricated with desired thicknesses of 5000Å polysilicon on 1000Å, 430Å, and 220Åsilicon dioxide on a silicon substrate. Test wafers were measured using a Nanospec and ellipsometer, and it was found that the wafers were actually 5100Å poly on 1100Å, 450Å, and 250Åoxide. Thus, all simulations use these measured thicknesses.

## 4.1 Lateral Interference

The matrix model accurately describes the reflected intensity provided that the pixel being looked at is homogeneous. Using a 756×244 pixel camera focused on a 4 inch diameter wafer, a pixel is approximately 130×400$\mu$m. A new system is currently in design which uses a 1000×1000 pixel camera, providing 100×100$\mu$m pixels. Given the sub-micron structures becoming increasingly common today, it is very unlikely that the entire pixel area is an homogeneous region, unless such a large region has been dedicated on the mask.

Patterning with a layer of photoresist results in parts of the wafer having an additional layer. Assuming that the photoresist is non-absorbing, the difference in height between two regions within the same pixel leads to lateral interference. In addition, the photoresist layer etches at a different rate than the polysilicon (depending on the selectivity of the etch process), further distorting the interferometry signal [17].

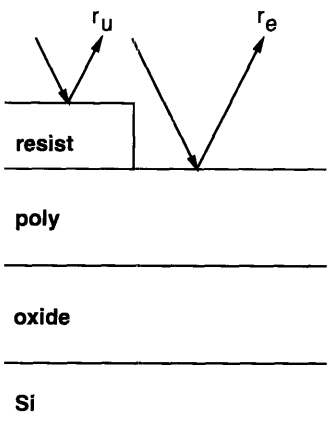


Figure 4-1: Patterned area model.

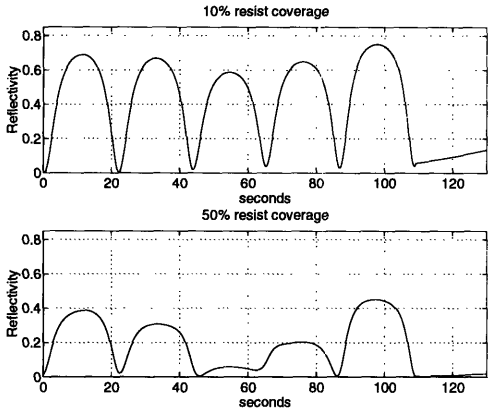


Figure 4-2: Simulated interferometry signals of areas with 10% and 50% resist coverage.

The lateral interference is caused by the phase difference resulting from two coherent light waves traveling through optical paths of different lengths (Fig. 4-1). Simulations for 10% and 50% resist coverage can be seen in Fig. 4-2. The amplitudes (not the intensities) of the two reflected light waves are summed to yield the resulting reflected intensity $R$:

$$R = |f_e r_e + f_u r_u|^2 \tag{4.1}$$

where $r_e$ and $r_u$ are the amplitudes of the reflected light, and $f_e$ and $f_u$ are the fractions of etched and unetched regions, respectively (note $f_e + f_u = 1$). A modified version of the matrix model [4] (App. A) was written to perform these simulations.

It is interesting to note that while the time signal of the 50% resist coverage plot in Fig. 4-2 appears very distorted, the FFT (Fig. 4-3) is nearly identical to the 10% coverage, with the exception of an additional peak at a lower frequency. This additional peak corresponds to the lower etch rate of the resist relative to the poly. A naive FFT method of computing etch rate falsely determines the etch rate of the process if there is more than 50% resist coverage. However, with knowledge of the resist coverage, the algorithm could recognize the resist contribution and ignore it when computing the polysilicon etch rate.

There is also diffraction at the step edges, but this effect can be ignored if the patterns are large relative to the wavelength being monitored [6]. Using a monitoring wavelength of 7534Å, the 1$\mu$m or smaller features may in fact cause diffraction effects. A thorough study of these effects is necessary, but is beyond the scope of this thesis.

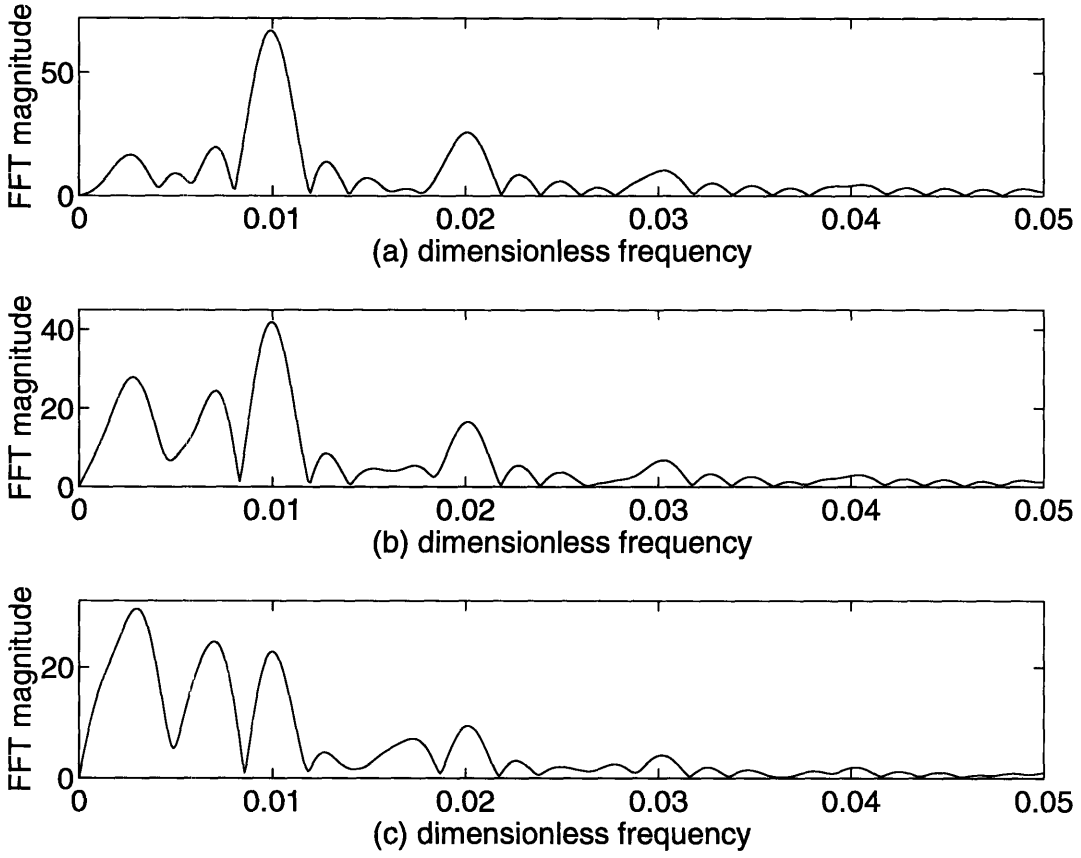


Figure 4-3: FFTs for simulated lateral interference effects. (a) is for 10% resist coverage, (b) for 30%, and (c) for 50%. Note how the presence of a lower FFT frequency (corresponding to the slower etch rate of resist) increases with the amount of resist.

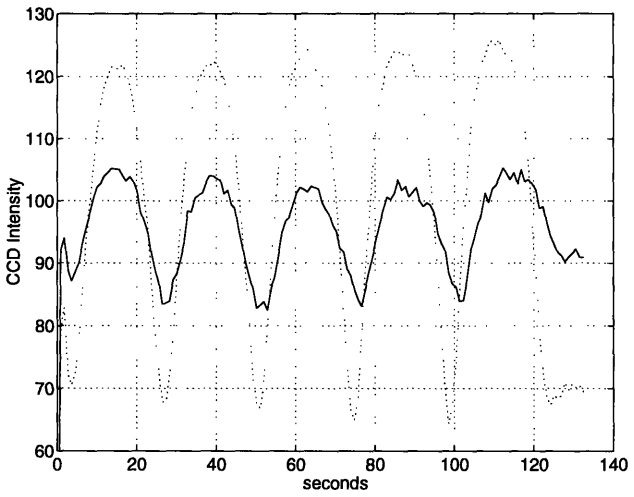


Figure 4-4: Experimental interferometry signal for 5100Å poly on 1100Å oxide with 1$\mu$m resist lines (solid). This is compared with the 100% poly signal (dotted).

As seen in Fig. 4-4, experimental measurements with a combination of resist and poly areas do not fit with the simulations in Fig. 4-2. It is speculated that we do not see lateral interference because the model assumes coherent light. The CCD interferometer uses diffuse light as its source; thus, two originating beams of light are not necessarily in phase. (An experiment using laser light aimed at this pattern could be performed in the near future to confirm this speculation). Another possible explanation for the departure from the predicted results is that photo-resist has anti-reflective agents added to the material. Thus, there is virtually no light being reflected from the resist region (i.e. $r_u = 0$). The results of this hypothesis produce a signal which looks similar to the 100% poly situation, but with an attenuated amplitude because there is less poly area to reflect the incident light. In fact, as shown in Fig. 4-4, the resulting signal is attenuated, and has approximately the same shape as the signal for pure poly, making a strong case for the second hypothesis. Regardless of the cause however, the CCD's departure from the lateral interference model is actually an advantage in that lateral interference effects can be ignored, thus making modeling simpler.

## 4.2 Non-Uniform Initial Thickness

Not all heterogeneous pixels result from areas of resist. A common situation is a uniform layer of polysilicon on top of a non-uniform layer of silicon dioxide, where "uniform" refers

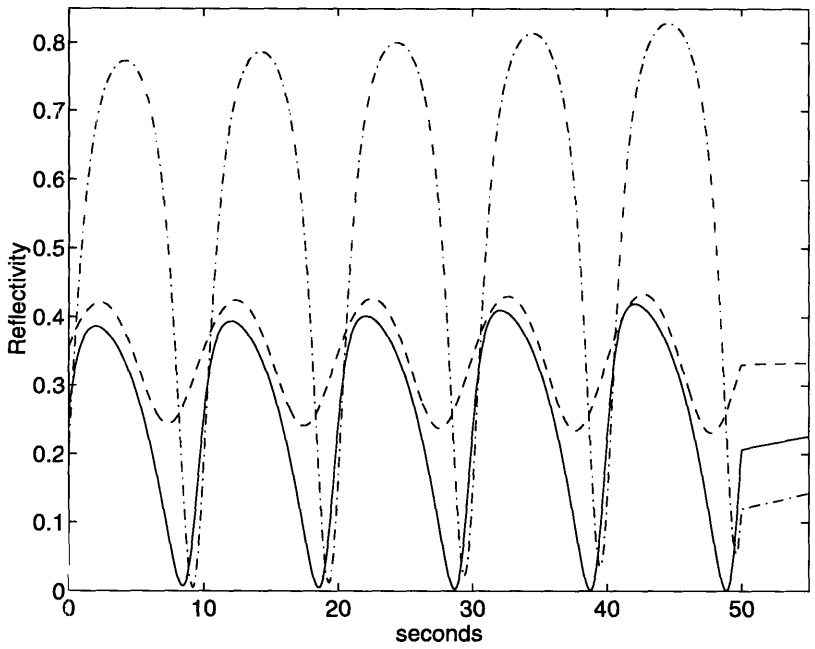


Figure 4-5: Simulated interferometry signal for initial oxide thickness of 1000Å (dash-dot) and 100Å (dashed), and the resulting signal (solid) from a region with 50% of each thickness.

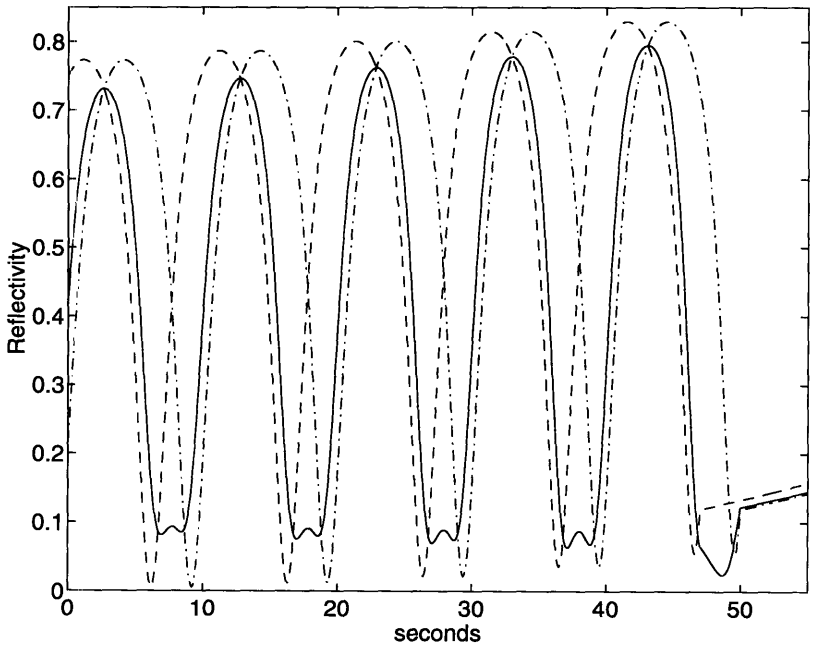


Figure 4-6: Interferometry signal for initial polysilicon thickness of 5000Å (dash-dot) and 4700Å (dashed), and the resulting signal (solid) from a region with 50% of each thickness.

to constant thickness. This occurs, for example, if a pixel includes areas of both the gate and the field oxide in an MOS process. Another situation leading to heterogeneous pixels is a non-uniform layer of polysilicon on top of a uniform layer of silicon dioxide. Figures 4-5 and 4-6 show interferometry simulations for regions of different initial oxide and polysilicon thicknesses, respectively. The solid lines are the simulated output, and the dashed lines are the individual signals from the two regions which were summed, assuming 50% of each area, to create the resulting output.

For large differences in initial polysilicon thickness, the complex sum is very distorted in the time domain, especially where a single minima turns into two minimums in Fig. 4-6. Regardless of this distortion, the magnitude of the FFT is virtually identical to an undistorted interferometry signal. Only the phase of the FFT is changed, because the distorted time-domain signal is essentially the sum of two time shifted sequences, and the frequency-domain effect of a time shift is a phase shift. Thus, the etch rate calculation (which relies on the magnitude of the FFT) of such a signal is not affected.

It is also interesting to note that the endpoint characteristic (the signal going "flat" due to the slower etch rate of oxide) seen in the simpler plots is still basically the same regardless of the added distortion.

## 4.3 Transition Layer

Another departure from the ideal model occurs because of the non-zero transition thickness layer between polysilicon and silicon dioxide [12]. It is believed that this transition layer may be 10 to 100Å thick.

For modeling purposes, the transition layer was simulated by assuming a linear change in index of refraction from polysilicon (3.73) and oxide (1.45). These index values were chosen based on a monitoring wavelength of 7534Å. Using the multilayer model, the transition layer was broken into 4 layers of indices (3.274, 2.818, 2.362, 1.906), resulting in an 8 layer model (vacuum, poly, transition $\times$ 4, oxide, silicon). A cross-section of this structure can be seen in Fig. 4-7.

The results of this simulation (Fig. 4-8) show that the sharp endpoint seen with an immediate transition is smoothed. The amount of smoothing increases as the transition layer thickness increases.

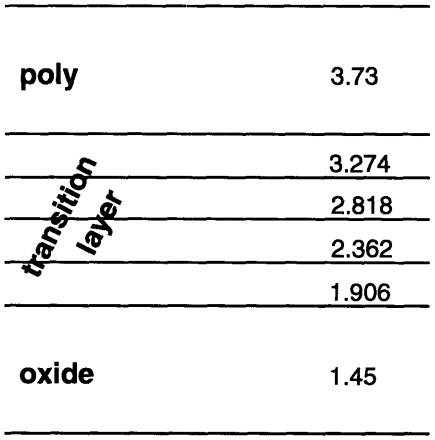


Figure 4-7: Transition layer cross-section

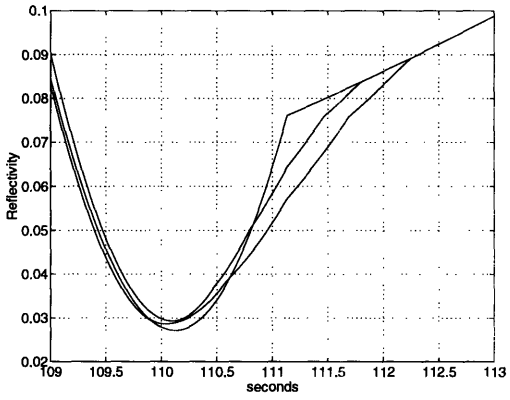


Figure 4-8: Close up of interferometry signal for polysilicon-oxide transition layer thicknesses of 0, 60, 100Å.

One relevant parameter that must be remembered is the sample rate of the data acquisition. These simulations were performed assuming a sampling rate of 10 Hz. However, the system used for the experiments acquires data at approximately 1.5Hz. It is possible that the transition layer thickness (provided it is reasonably small) has no effect on the interferometry signal. Assuming a worst case scenario of a transition layer of 100Å and an etch rate of 2000Å/min, it would take 3 seconds to etch through the transition layer. And assuming a best case with a transition layer of 10Å and an etch rate of 6000Å/min, it would take 0.1 seconds. The worst case scenario, even with the system's relatively slow rate acquisition, reveals the smoothing effects of the transition layer. However, these effects do not appear in the best case.

There is another transition layer which exists between the top surface of the wafer and the vacuum. The effect of this transition layer is a slight phase shift due to additional refraction. Fortunately, this effect is constant throughout the entire process and thus does not need to be included in additional process control modeling.

## 4.4 Bandpass Filter

All models in this thesis assume that the bandpass filter on the CCD camera is perfect; i.e., only light of *exactly* 7534Å (or 4800Å) passes through. The 7534Å bandpass filter used in these experiments has a bandwidth of 320Å. A simulation using a model which assumes that 25% of light passes through at 7374Å, 50% at 7534, and 25% at 7694Å was performed using

a method similar to that of patterned areas: the total reflected intensity is the square of the sums of each reflected amplitude. The resulting interferometry signal was only slightly different than all the other interferometry signals, and not different enough to have any impact on etch rate or endpoint calculations. It is therefore concluded that the bandwidth of this filter is small enough such that the filter can be treated as an "ideal" bandpass filter.

## 4.5 Non-Normal Incidence

Thus far, these models have ignored incidence angle and assumed it to be zero. The result of non-normal incidence is light scattering, which ultimately can be considered an additional noise source. Below are some of the reasons why non-normal incidence occurs.

There is actually non-normal incidence when looking at any pixel not in the center of the wafer, but the angle is still small enough to be ignored. The system used in the experiments uses 4 inch wafers which are approximately 10 inches away from the camera lens. Therefore, the incident angle at the edge of the wafer is approximately 11°. Industry is currently using 8 or 10 inch wafers, which may or may not cause an increase in the incident angle depending on the camera's positioning above the wafer.

In addition, surface roughening causes non-normal incidence upon reflection because the surface may not be flat, leading to a decrease in signal amplitude.

Finally, we have assumed that step and profile coverage is perfect rather than gradual. Gradual slopes have similar light scattering and signal attenuating effects as due to surface roughening.

## 4.6 Index of Refraction

In addition to the index of refraction dependencies discussed in Sect. 2.3.1, factors such as temperature and doping concentration can have effects on the index of refraction. Fortunately, given the relatively short wavelength (7534Å) and the relatively low temperature ($<$ 50°C) of our process, these effects can be ignored. Sturm and Reaves [18] found that for infrared wavelengths and shorter there is less than a 2% change in the index of refraction of silicon over temperatures from room temperature to 700°C. The index of refraction of polysilicon is identical to that of silicon, and at wavelengths relevant to our research, is not dependent upon doping concentration [5].

## 4.7 Modeling Results

As a "sanity check" for using the CCD interferometry signal for etch rate and endpoint, some simple comparisons were performed. We can compare the measured etch rate to the "predicted" etch rate, where the predicted etch rate is 5100Å times the measured endpoint time. Then a percentage agreement is computed as 100 times the difference between the measured and the predicted etch rates, all divided by the measured etch rate. The results of this computation, in a spatial arrangement as found on the wafer, are in Table 4-9, and show that the measured and predicted etch rates were within better than 2.5% of one another. Similar calculations were performed for endpoint agreement and initial thickness agreement.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.87 | 1.16 | 0.17 | -0.43 | -0.07 | 2.38 | |
| 2.32 | 1.41 | -0.64 | -0.27 | -0.08 | 0.39 | 1.25 | 2.09 |
| 0.78 | -0.64 | 0.72 | -0.73 | -0.55 | -0.73 | -0.43 | -0.43 |
| -0.43 | -0.43 | 0.08 | 0.25 | 0.25 | -0.55 | -0.91 | -0.24 |
| -0.68 | -0.43 | -0.08 | -0.73 | -0.55 | -0.73 | -1.11 | -1.30 |
| 0.89 | -0.24 | -1.11 | -0.08 | -0.08 | -0.27 | -0.01 | 0.02 |
| 1.07 | 2.39 | -0.07 | -0.24 | -1.09 | -0.68 | 1.24 | 1.77 |
| | 0.35 | 1.50 | 2.09 | 1.82 | 1.82 | 0.51 | |

Figure 4-9: Percentage difference between measured etch rate and "predicted" etch rate for all 60 die on the wafer.

Wafer maps of endpoint versus location (Fig. 4-10) and etch rate versus location (Fig. 4-11) show that, as expected [19], the wafers are etched with good azimuthal but poor radial uniformity.

## 4.8 Conclusion

This chapter has discussed several non-ideal wafer characteristics which complicated the originally presented models. Addressing these issues and including updated models allows the algorithms to perform properly under some non-ideal conditions. The next chapter introduces the hardware and software on which these algorithms are implemented.

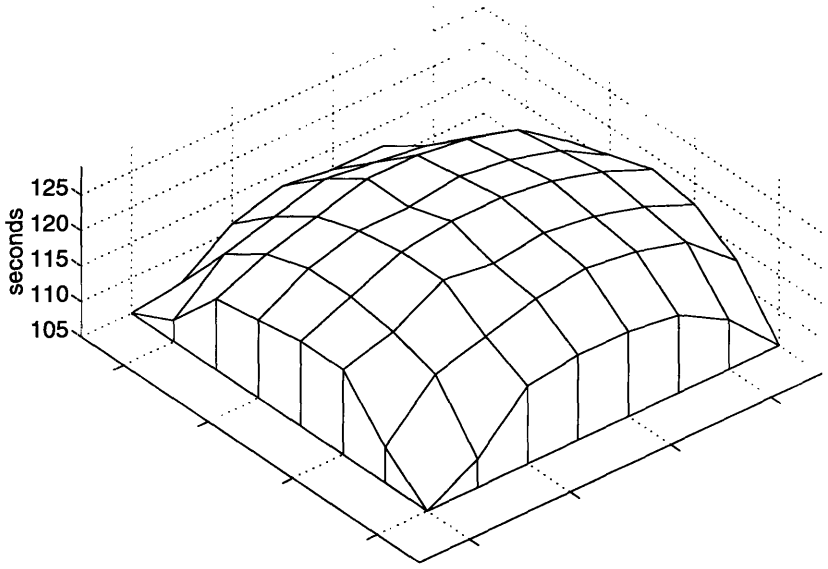


Figure 4-10: Time of endpoint for all all 60 die on a wafer.

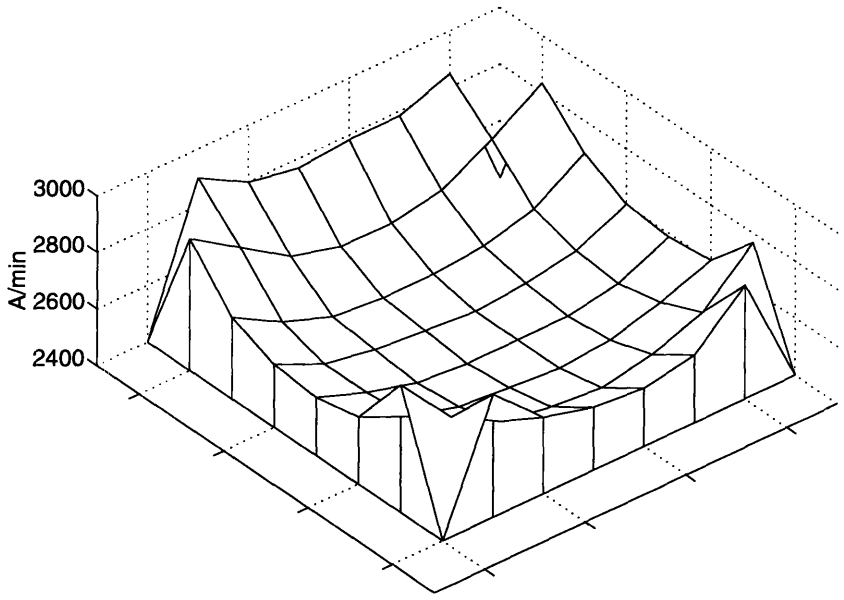


Figure 4-11: Etch rate for all 60 die on a wafer.

# Chapter 5

# Experimental Setup

## 5.1 Applied Materials Precision 5000 Plasma Etcher

All experiments were performed using an Applied Materials Precision 5000 Plasma Etcher (AME-5000). This machine has a 50mm viewport at the top of the vacuum chamber, thus making it convenient for *in-situ* monitoring. As mentioned previously, it operates in a rotating magnetic field, thereby increasing circumferential etching uniformity.

## 5.2 Old System

The original system for full wafer interferometry was built using an Electrim EDC-1000HR CCD camera and an 80386-based microcomputer [7]. The camera was situated over the viewport of the AME-5000 and was able to focus on the entire 100mm wafer using a 16mm focal length lens.

The EDC-1000HR camera has a resolution of 756×244 pixels with 8 bit pixel depth. The software on the 386 was capable of acquiring at most 2 frames per second due to the bandwidth limitations of a DOS machine's bus. The control program spent all of its time acquiring data, leaving no resources available for analysis. Thus, all analysis was done post-process using images that were saved during the process.

The analysis program displays the first image of the series and allows the user to select certain pixels for analysis using the mouse. Plots of intensity versus time, as well as an FFT and etch rate for each selected pixel are displayed.

This system provides post-process diagnostic tools. It lacks the capability of analysis

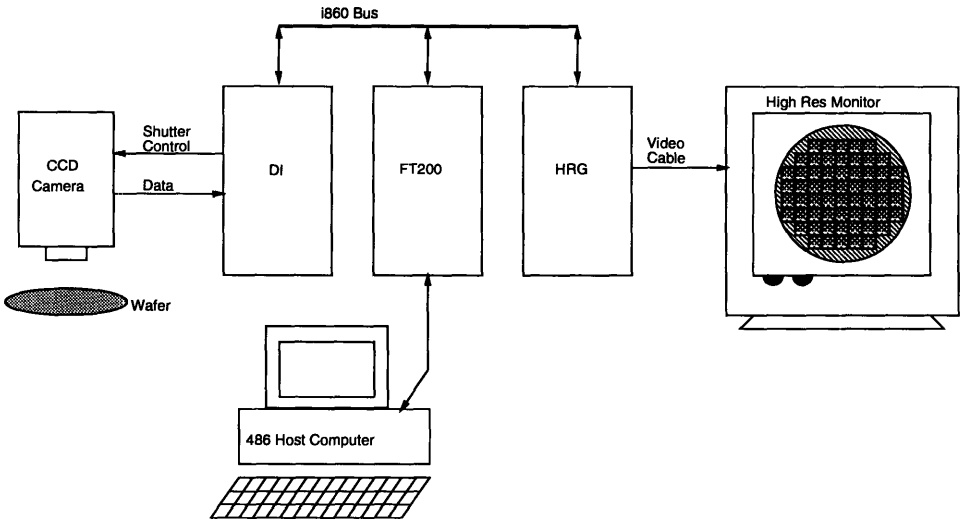


Figure 5-1: Simplified diagram of the acquisition hardware.

during a process, which is a necessary step towards real time control.

## 5.3 Hardware Overview

A new system was installed as an upgrade to the existing system. The powerful features of the new system, most notably increases in resolution, dynamic range, and bandwidth, allow the system to be used as a research tool to probe the usefulness and limitations of a real time control apparatus. The new system was not meant to be an end-user product. This new system consists of a Hamamatsu C4742 CCD camera and Alacron acquisition, processing, and display boards within an 80486-based microcomputer. A simplified diagram of the setup is shown in Fig. 5-1.

The Alacron system consists of three boards: an acquisition board (the DI), a processing board with two Intel i860 RISC processors, and a display card (the HRG) which all share a common bus in addition to being on the 486's ISA bus. A maximum of 56 MB of DRAM can be on the Alacron boards when both the DI and HRG are installed. (Otherwise a maximum of 64 MB can be installed). While 56 MB may sound like a lot of memory, a 1000×1000 image with 8 bit pixels takes up approximately 1 MB. And at real time video rates of 30 frames per second (fps), this amounts to just under 2 seconds of video. Fortunately, real time video rates are not necessary for CCD interferometry measurements.

The DI card performs the data acquisition. It takes in the data lines (up to 32 bits) and

control lines (frame valid, line valid, and pixel clock), and controls a direct memory access (DMA) transfer of the data to a specified target address. It uses FIFOs and a word size converter to send the data over a 64 bit bus in a burst transfer, providing a higher DMA bandwidth.

The HRG (high resolution graphics) card allows data to be displayed on a second monitor as it is acquired, rather than sending it over the ISA bus to be displayed on the VGA monitor. There are built in routines to handle resolutions of 640×480, 800×600, 1024×768, and 1152×900. User-customized resolutions are also possible by directly accessing the Hitachi video controller chip used by the HRG.

The processor card (referred to as the FT200) contains dual i860 chips and up to 16 MB of memory on the board. Up to two daughter cards, with up to 24 MB each, may also be installed as add-ons. Memory management support for a shared processor environment is provided, along with a real time clock (RTC) circuit.

From a marketing standpoint, the features of the new equipment provide substantial benefits. The camera has over 1000×1000 resolution, allowing finer patterning detail of the wafer. However, from a development standpoint, the 1000×1000 resolution requires nearly four times as much memory to store an image. It also does not offer nearly enough resolution to be more useful than a lower resolution camera; one pixel of the image corresponds to approximately $100\mu m^2$, which remains very large compared to the sub-micron structures being fabricated today. It would be interesting to use a larger lens (perhaps greater than 50mm focal length) to focus the CCD on one $1cm^2$ die, where each pixel would correspond to $1\mu m^2$, however this experiment has not yet been performed. In addition, it has 10 bit pixels, providing 1024 gray levels compared with the 256 gray levels (8 bits) of most cameras. The higher dynamic range is achieved using Peltier cooling, which reduces the dark current which prevents some CCD noise. However, 10 bit pixel depth means that an image requires twice as much memory if the pixel is stored in 16 bit memory blocks. An alternative is a bit-packing routine which packs 10 bit numbers aligned to 16 bit blocks into 8 bit blocks which cross byte boundaries. The bit packing routine is easily written, but it slows down the memory transfer.

## 5.4 Software Design

### 5.4.1 Overview

There are two main modes of operation for the FT200, stand alone mode and slave processor mode. In stand alone mode, a program is written entirely for the i860. At run time, the program is loaded and executed on the i860 by a single thread host environment. The i860 programs are written on the 486, and cross-compiled by the Portland Group i860 compiler. There are two programs necessary for slave processor mode operation: one which contains the routines to be run on the i860, and one which runs on the 486 and controls the execution of those i860 routines. In this mode, the i860 code is compiled by the Portland Group compiler, while the control program is compiled by a DOS compiler such as Microsoft C 7.0. At run time, the controlling program loads the i860 program to the i860, and selectively calls routines.

Alacron offers several libraries for use with the i860. These include a Vector Library for vector math and DSP routines, a scientific image processing library, and the Intel Graphics Library.

### 5.4.2 Module Descriptions

The flow of a generic camera control program is as follows:

1. Initialization.
2. Computer triggers the camera to open the shutter for a specified amount of time.
3. The camera automatically begins clocking out the data upon the closing of the shutter.
4. The input card buffers the incoming data and controls the DMA to a preset destination address.
5. The controlling program polls the input card to check when no more data is being transferred.
6. If desired, all steps after the initialization are repeated.

The analysis software adds a layer on top of this generic model. It begins by taking a picture of the wafer. A user can then select points to analyze on the wafer by moving the mouse to those points and clicking. The user then acknowledges that all points have been selected, and the etching and acquisition begin.

The implementation of the above model on the Alacron hardware is described below. The full code can be found in Appendix B.

### DI and HRG Initialization.

Initialization is performed by the subroutine `dcl_init()`. The main tasks of this routine are:

- Start the RTC circuitry, and set the clock to a frequency of 1000Hz.
- Set the HRG to the proper (custom) resolution.
- Set the HRG display palette to 256 gray scales.
- Set the DI input word size to 8 bits (for 256 gray scales) or 16 bits (of which only 10 contain data, resulting in 1024 gray scales).
- Enable DMA and set the destination address to the HRG display memory.

### Camera Control.

The Hamamatsu camera can be controlled via a single TTL level input. When this input is logically high, the shutter is closed. When this signal is low, the shutter is open. Thus, a picture can be taken by opening the shutter, waiting a desired amount of time (on the order of 25 milliseconds), and then closing the shutter.

### Acquisition and Display.

Data acquisition begins automatically when the shutter closes. The camera begins sending data over the parallel RS-422 cable, and the DI (provided it has been properly initialized) begins buffering the data and overseeing DMA transfers of the data. The DMA destination address is set to the base address of the display, so that the images are displayed as they are acquired. The maximum acquisition rate is limited primarily by the CCD transfer clock at 10MHz and the shutter exposure time. Maximum acquisition rates of over 8 frames per second can be sustained.

A tricky part of the initialization module handles setting a custom resolution for the display. The easy way to acquire and display an image is to set the DMA destination to be some arbitrary memory address, and then, pixel by pixel, copy the data from this memory location to the display card. In this manner, extra pixels can be skipped or added to

memory to disk. The constraint here is memory size. As mentioned above, there are only 56MB of memory on the Alacron boards, providing space for less than 55 images. Even at a barely acceptable frame rate of 1Hz, this only allows 55 seconds worth of storage for a process that usually lasts close to two minutes.

One thought is to use an old queuing theory exercise to solve this problem. Imagine a cup with a small hole in the bottom. When water is poured into the top of the cup, it begins to trickle out through the hole in the bottom. Now compare this cup to memory. Writing to memory using `memcpy()` is like pouring water in, and writing to disk is like the water slowly trickling out. Basically, the memory is acting like a FIFO buffer. Assume that the memory can normally hold $M$ images, and that one image can be written to memory in $A$ seconds, and one image can be written to disk in $B$ seconds. Under ideal conditions, $M * B/(B - A)$ images can now be saved before the memory overflows. Unfortunately, given that $A$ is around 0.05 seconds, $B$ is around 3 seconds, $B/(B - A)$ is very close to 1, so almost no benefit is obtained.

In addition to speed, we are also concerned with efficient space usage during storage. It is convenient to store 8 bit numbers, as each value is stored in a byte of memory. However, 10 bit numbers must be stored in memory as 16 bit numbers with the highest 6 bits set to zero. As mentioned previously, bit-packing and -unpacking routines were written to perform these tasks. The code segment to pack the bits appears below.

```
for (i=0, j=0; i<IMAGE_SIZE; i+=4, j+=5)
{
  output[j]   =   input[i]   & 0x00ff;
  output[j+1] = ((input[i]   & 0x0300)>>8) + ((input[i+1] & 0x003f)<<2);
  output[j+2] = ((input[i+1] & 0x03c0)>>6) + ((input[i+2] & 0x000f)<<4);
  output[j+3] = ((input[i+2] & 0x03f0)>>4) + ((input[i+3] & 0x0003)<<6);
  output[j+4] = ((input[i+3] & 0x03fc)>>2);
}
```

### Signal Processing Routines.

The Alacron vector library provides all the necessary tools to compute a windowed etch rate. The algorithm to compute etch rate uses (3.1). Matlab code to compute the fundamental frequency of an interferometry signal stored in a variable called `data` might look like:

```
[y, f] = max(abs(fft(data - mean(data), 1024)));
```

where y contains the max FFT value and f contains the index of that value. Note that f divided by the FFT length (in this example it is 1024) is the fundamental frequency referred to in (3.1). This Matlab code performs the following steps in a simple, easy to program, manner:

1. Compute the mean value of the data.
2. Remove the bias from the data by subtracting the mean value from the data.
3. Take the 1024 point FFT of this unbiased data.
4. Compute the absolute value (magnitude) of the FFT coefficients.
5. Determine the index of the maximum FFT coefficient. This is the fundamental frequency.

Writing the equivalent code for use on the Alacron system is not as straightforward. In order to achieve maximum performance out of the i860 processors, special C functions are provided from the Alacron vector library. Unfortunately, these functions have very strict constraints on their arguments (for example the input and output of the FFT (`rfftb()`) must be quad-word aligned), and thus the code is not as simple as the Matlab equivalent.

```
meanv (data, 1, &mean, datalength);              /* compute the mean */
mean = -mean;
vsadd (data, 1, &mean, nobias, 1, datalength);   /* subtract it out   */
zeropad (nobias, datalength, FFTLEN);            /* zeropad the data  */
rfftb (nobias, fftresult, FFTLEN, 1);            /* compute the FFT   */
rfftsc (fftresult, FFTLEN, 2, 1);                /* unpack the result */
cvmags (fftresult, 2, sqmagfft, 1, 1024);        /* mag. of result    */
maxv (sqmagfft, 1, &maxvalue, &maxindex, FFTLEN); /* get max index     */
```

**Timer Functions.**

The RTC circuitry on the FT200 provides a real time clock with a programmable frequency, which was set to 1000Hz. Routines were written to pause for a given number of milliseconds, and to return a time stamp for time tracking purposes.

# Chapter 6

# Conclusion

This thesis has discussed the use of a CCD camera as an *in-situ* monitor of plasma etch processes. Each pixel of the camera acquires an interferometry signal similar to that obtained from a laser interferometer. The interferometry signal is processed to yield etch rate and endpoint measurements, which can be used for diagnostics or for real time control. The etch rate and endpoint results account for non-ideal wafer characteristics through careful modeling.

## 6.1 Future work

There is still a large amount of work that could be done as a continuation of this research. Section 2.4 discussed the effects of a rising OES signal on the interferometry signal. It is possible that the endpoint characteristic visible in an ideal interferometry signal will be hidden. It is suggested that future research investigate the use of an endpoint detector which combines both data for a more robust endpoint.

Section 3.3 explored the use of a tunable wavelength filter with the CCD system to measure film thickness. We believe a film thickness measurement is possible because it closely follows the algorithms used by the Nanospec to measure film thickness. We purchased a Varispec filter, made by Cambridge Research Instruments, to perform these measurements, but did not have the necessary lenses. A film thickness measurement in real time would be invaluable, justifying continued effort on using the Varispec.

Diffraction effects due to monitoring a patterned area with a wavelength of comparable size to the features were introduced in Sect. 4.1. A more detailed study of these effects,

along with their impact on the models, is needed. In addition, reasons why the experimental results did not agree with the lateral interference model were theorized. An experiment using laser interferometry aimed at the patterned area should be performed to test these theories.

There are many items that could be improved upon in the software (Sect. 5.4). These are enumerated below.

- Add a Windows GUI for user-friendly operation.
- Investigate signal integrity in the cable connecting the camera to the DI. During acquisition, several lines "loose sync" and are skewed by a few pixels. The stored images, and thus the data, are not affected, so it is assumed that this is a problem in the display card being affected by the noise of the data transfer.
- Rewrite some time-critical components (e.g. a down sample routine) in machine language for performance improvements.

Looking forward, there are many image processing techniques that could be implemented. One idea is to use a reference image which can be "subtracted out" of each image in order to "remove" the wafer's pattern from the image. This having been done, intensity gradients across a wafer could yield immediate feedback on etch rate uniformity.

A second image processing tool could be used for wafer alignment. While processing many wafers in series, it may be desired to monitor the same exact point on each wafer. Naively choosing the same pixel coordinate does not guarantee that the same point on the wafers will be monitored because each wafer is not necessarily in the same exact location in the chamber. Object recognition could be performed to locate a certain reference point on each wafer, and the monitoring point coordinates could be relative to that point.

While the CCD camera has already provided a significant amount of information regarding the control of the plasma etch, there is a significant amount to learn. The CCD camera provides a research tool which will allow much of this work to be accomplished in the future.

# Bibliography


[1] S. Wolf and R. N. Tauber, *Silicon Processing for the VLSI Era, Volume 1.* Sunset Beach, CA: Lattice Press, 1986.

[2] S. Ha, *On-line control of process uniformity using categorized variabilities.* PhD thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, 1993.

[3] P. H. Berning, "Theory and calculations of optical thin films," *Physics of Thin Films*, vol. 1, p. 69, 1963.

[4] O. S. Heavens, *Optical Properties of Thin Solid Films.* New York: Dover Publications, 1955.

[5] D. F. Edwards, *Handbook of Optical Constants of Solids*, pp. 547–569. Orlando: Academic Press, 1985.

[6] P. A. Heimann and R. J. Schutz, "Optical etch-rate monitoring: Computer simulation of reflectance," *J. Electrochem. Soc.*, vol. 131, no. 4, pp. 881–885, 1984.

[7] T. J. Dalton, W. T. Conner, and H. H. Sawin, "Interferometric real-time measurement of uniformity for plasma etching," In press, *J. Electrochem. Soc.*, 1994.

[8] V. Patel, W. Kosonocky, S. Ayyagari, and M. Patel, "Application of thermal imaging methodology for plasma etching diagnosis," *Process Module Metrology, Control, and Clustering, Proc. SPIE*, vol. 1594, pp. 204–208, 1991.

[9] A. G. Nagy, "Radial etch rate nonuniformity in reactive ion etching," *J. Electrochem. Soc.*, vol. 131, no. 8, pp. 1871–1875, 1984.

[10] D. S. Grimard, J. Fred, L. Terry, and M. E. Elta, "In situ wafer monitoring for plasma etching," *Dry Processing for Submicrometer Lithography, Proc SPIE*, vol. 1185, pp. 234–247, 1989.

[11] D. S. Grimard, J. Fred, L. Terry, and M. E. Elta, "Theoretical and practical aspects of real-time Fourier imaging," *Advanced Techniques for Integrated Circuit Processing, Proc SPIE*, vol. 1392, pp. 535–542, 1990.

[12] H. L. Maynard and N. Hershkowitz, "New alternative graphing methods for thin-film interferometry data," In press, *IEEE Transactions on Semiconductor Manufacturing*, 1992.

[13] P. F. Cox and A. F. Stalder, "Measurement of Si epitaxial thickness using a Michelson interferometer," *Journal of Electrochemical Society*, vol. 120, no. 2, pp. 287–292, 1973.

[14] S. A. Henck, "In situ real-time ellipsometry for film thickness measurement and control," *Journal of Vacuum Science and Technology*, vol. 10, no. 4, pp. 934–938, 1992.

[15] F. G. Boebel and H. Moller, "Simultaneous in situ measurement of film thickness and temperature by using multiple wavelengths pyrometric interferometry (MWPI)," *IEEE Transactions on Semiconductor Manufacturing*, vol. 6, no. 2, pp. 112–118, 1993.

[16] Z.-H. Zhou, I. Yang, F. Yu, and R. Reif, "Fundamentals of epitaxial silicon film thickness measurements using emission and reflection Fourier transform infrared spectroscopy," *Internal Publication*, 1993.

[17] P. A. Heimann, "Optical etch-rate monitoring using active device areas: Lateral interference effects," *J. Electrochem. Soc.*, vol. 132, no. 8, pp. 2003–2006, 1985.

[18] J. C. Sturm and C. M. Reaves, "Silicon temperature measurement by infrared absorption: Fundamental processes and doping effects," *IEEE Transactions on Electron Devices*, vol. 39, no. 1, pp. 81–88, 1992.

[19] D. Boning, S. Ha, and E. Sachs, "On-line control of uniformity in single-wafer plasma etch processes," *Proceedings SRC TECHCON 93*, pp. 19–21, 1993.

# Appendix A

# Matrix Model for Multilayer Structures

```
/* this file contains a multilayer model based on Heavens, pp. 74-80. */

#include <stdio.h>
#include <math.h>

#define MAXLAYERS 25
#define pi 3.141592653595

#define sq(A) ( (A) ? ( (A < 0) ? pow(-1 * A, 2.0) : pow(A, 2.0) ) : 0 )

/* I can't get the default value thing to work */
#define inputd(A, B, C) printf(A, B); scanf("%d", C)
#define inputf(A, B, C) printf(A, B); scanf("%lf", C)

main(argc, argv )
int argc;
char **argv;
{
  FILE *fp;
  int m, layers = 4;

  double d[MAXLAYERS], n[MAXLAYERS], k[MAXLAYERS], er[MAXLAYERS];

  double lambda = 7534.0, samples=10;

  double alpha[MAXLAYERS], gamma[MAXLAYERS];

  double g[MAXLAYERS], h[MAXLAYERS];
```

```
double p[MAXLAYERS], q[MAXLAYERS], r[MAXLAYERS], s[MAXLAYERS];
double t[MAXLAYERS], u[MAXLAYERS], v[MAXLAYERS], w[MAXLAYERS];

double p1[MAXLAYERS], q1[MAXLAYERS], r1[MAXLAYERS], s1[MAXLAYERS];
double t1[MAXLAYERS], u1[MAXLAYERS], v1[MAXLAYERS], w1[MAXLAYERS];

double R;

/* default values */
/* vacuum */
n[0] = 1;           /* index of refraction */
k[0] = 0;           /* extinction coefficient */
/* poly */
d[1] = 5000;         /* film thickness */
n[1] = 3.73;
k[1] = 0.02;
er[1] = 6000;        /* etch rate, Angstroms/min */
/* oxide */
d[2] = 1000;
n[2] = 1.45;
k[2] = 0;
er[2] = 1000;
/* silicon substrate */
n[3] = 3.73;
k[3] = 0.01;
d[3] = 1000;

/* Stupid (but necessary) user interface */

printf("This program assumes you have a layer of vacuum followed by a number\n");
printf("of films, with the last layer being the substrate.  Please enter the\n");
printf("number of layers, including the vacuum and the substrate.\n\n");

inputd("Layers [%d]? ", layers, &layers);
if (layers < 3 || layers > 25)
{
  printf("You must have between 3 and 25 layers\n");
  exit(0);
}

/* more defaults */
for (m=4; m<layers; m++)
{
  n[m] = 1.5;
  k[m] = 0;
  d[m] = 1000;
  er[m] = 1000;
}
```

```
inputf("Wavelength [%.0f]: ", lambda, &lambda);
lambda *= 1E-10;

inputf("Samples per second [%.0f]: ", samples, &samples);

for ( m=1; m<layers; m++ )
{
  if (m<layers-1)
    printf("Film %d:\n", m);
  else
    printf("Substrate:\n");

  inputf("  Index of refraction [%1.2f]? ", n[m], &n[m]);
  inputf("  Extinction coefficient [%1.2f]? ", k[m], &k[m]);
  if (m<layers-1)
  {
    inputf("  Film thickness in angstroms [%.0f]? ", d[m], &d[m]);
    d[m] *= 1E-10;
    inputf("  Etch rate in angstroms/minute [%.0f]? ", er[m], &er[m]);
    er[m] / samples * 60E10;  /* adjust for angstroms/sample-rate */
  }
}

fp = fopen("output", "w");

/* Let the calculations begin! */

/* Loop over the layers to be etched.  Begin by etching layer 1,
   then etch each subsequent layer until only the substrate is left. */
while (layers > 2)
{
  for (m=1; m<layers; m++)
  {
    g[m] = ( sq(n[m-1]) + sq(k[m-1]) - sq(n[m]) - sq(k[m]) ) /
           ( sq(n[m-1] + n[m]) + sq(k[m-1] + k[m]) );
    h[m] = 2 * ( n[m-1] * k[m] + n[m] * k[m-1] ) /
           ( sq(n[m-1] + n[m]) + sq(k[m-1] + k[m]) );
  }

  /* etch the top layer */
  for ( ; d[1] > 0 ; d[1] -= er[1] )
  {
    for ( m=1; m<layers; m++ )
    {
      if ( m < layers-1 )
      {
        alpha[m] = 2*pi*k[m]*d[m] / lambda;
```

```
    gamma[m] = 2*pi*n[m]*d[m] / lambda;
  }

  if (m > 1)
  {
    p[m] = exp( alpha[m-1] ) * cos( gamma[m-1] );
    q[m] = exp( alpha[m-1] ) * sin( gamma[m-1] );
    r[m] = exp( alpha[m-1] ) *
          ( g[m] * cos( gamma[m-1] ) - h[m] * sin( gamma[m-1] ) );
    s[m] = exp( alpha[m-1] ) *
          ( h[m] * cos( gamma[m-1] ) + g[m] * sin( gamma[m-1] ) );
    t[m] = exp( -1 * alpha[m-1] ) *
          ( g[m] * cos( gamma[m-1] ) + h[m] * sin( gamma[m-1] ) );
    u[m] = exp( -1 * alpha[m-1] ) *
          ( h[m] * cos( gamma[m-1] ) - g[m] * sin( gamma[m-1] ) );
    v[m] = exp( -1 * alpha[m-1] ) * cos( gamma[m-1] );
    w[m] = -1 * exp( -1 * alpha[m-1] ) * sin( gamma[m-1] );
  }
}

p1[1] = 1;
q1[1] = 0;
t1[1] = g[1];
u1[1] = h[1];
r1[1] = g[1];
s1[1] = h[1];
v1[1] = 1;
w1[1] = 0;

for (m=2; m<layers; m++)
{
  p1[m] = p1[m-1] * p[m] - q1[m-1] * q[m] +
          r1[m-1] * t[m] - s1[m-1] * u[m];
  q1[m] = q1[m-1] * p[m] + p1[m-1] * q[m] +
          s1[m-1] * t[m] + r1[m-1] * u[m];
  r1[m] = p1[m-1] * r[m] - q1[m-1] * s[m] +
          r1[m-1] * v[m] - s1[m-1] * w[m];
  s1[m] = q1[m-1] * r[m] + p1[m-1] * s[m] +
          s1[m-1] * v[m] + r1[m-1] * w[m];
  t1[m] = t1[m-1] * p[m] - u1[m-1] * q[m] +
          v1[m-1] * t[m] - w1[m-1] * u[m];
  u1[m] = u1[m-1] * p[m] + t1[m-1] * q[m] +
          w1[m-1] * t[m] + v1[m-1] * u[m];
  v1[m] = t1[m-1] * r[m] - u1[m-1] * s[m] +
          v1[m-1] * v[m] - w1[m-1] * w[m];
  w1[m] = u1[m-1] * r[m] + t1[m-1] * s[m] +
          w1[m-1] * v[m] + v1[m-1] * w[m];
}
```

```
      R = ( sq(t1[layers-1]) + sq(u1[layers-1]) ) /
          ( sq(p1[layers-1]) + sq(q1[layers-1]) );
      fprintf(fp, "%f\n", R);
    }

    /* A layer has just cleared.  Prepare to etch the next layer by
     * moving all the remaining layers up: if we had air|poly|oxide|Si,
     * rearrange all the constants so that we now have air|oxide|Si
     */

    layers--;
    for(m=1; m<layers; m++)
    {
      n[m] = n[m+1];
      k[m] = k[m+1];
      d[m] = d[m+1];
      er[m] = er[m+1];
    }

  }
  fclose( fp );
}
```

# Appendix B

# Data Acquisition and Analysis Software

```
/*****************************************************************************
|
|  ccdctrl.c  -  The CCD control program
|
|  This program runs on the 486 and controls the i860 functions
|
|  Written by: Jonathan Claman
|
|*****************************************************************************/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <stddef.h>
#include <malloc.h>
#include <allib.h>
#include <graph.h>
#include <dos.h>
#include <mouse.h>

#define DEV        0
#define I860PROGRAM    "ccd"
#define STACKSIZE    100000L

#define NUMPICS        100
#define IMAGE_SIZE    1040*1024
#define SMALL_X        520
#define SMALL_Y        480

float timestamp (void);
ADDR i860malloc (long int);
```

```
main ()
{
    FILE *fp;
    long foo;
    int i, j, k;
    int x, y, b, oldx=0, oldy=0, oldb=0, numpts=0;
    char tmp[100];
    float time1, time2;
    ADDR ram860, small, points;
    char __huge *ram486;

    if((i = CheckMouse()) == 0)
    errexit ("No mouse found -- you need to run \"mouse\".");

    if (alopen (DEV) ≠ 0)
    errexit ("can't open AL860 device %d\n", DEV);
    aldev (DEV);

    if (almapload (I860PROGRAM, STACKSIZE) ≠ 0)
    errexit ("can't load %s\n", I860PROGRAM);

    ram860 = i860malloc ((long) IMAGE_SIZE);
    small  = i860malloc ((long) IMAGE_SIZE);

    if ( (ram486 = _halloc((long) SMALL_X, sizeof(char))) == NULL )
    errexit ("Can't malloc 486 mem %d\n", i);

    if (!_setvideomode(_MAXRESMODE) ) {
    printf("Error setting video mode :-( \n");
    exit(0);
    }

    alcall ( aladdr("_dcl_init"), 0);
    alwait ();

    alcall ( aladdr("_acquire_and_display_once"), 1, ram860);
    alwait ();
    alcall ( aladdr("_acquire_and_display_once"), 1, ram860);
    alwait ();
//    alcall ( aladdr("_shrink_image"), 2, small, ram860);
//    alwait ();
//    alcall ( aladdr("_display_small_windowed_image"), 3, small, 0L, 0L);
//    alwait ();

    LimitCursor(HORIZ, 10, 1039);
    LimitCursor(VERT,  8, 999);
    i = 3;
    _settextposition(i++,0);
```

```
    _outtext("Analysis points");
    while(!kbhit())
    {
    b = CheckPosition(&x, &y);
    _settextposition(0,0);
    sprintf(tmp, "Buttons: %d, mouse at %d col and %d row    ", b, x, y);
    _outtext (tmp);
    alcall ( aladdr("_mouse_handler"), 6, (long) b, (long) x, (long) y,
                        (long) oldb, (long) oldx, (long) oldy);
    alwait ();
    if (b == 1 && oldb ≠ 1)
    {
       _settextposition(i++, 0);
       sprintf(tmp,"%d,%d\n", x, y);
        _outtext(tmp);
       numpts++;
    }
    oldb = b;
    oldx = x;
    oldy = y;
    }
    _getch();

    alcall ( aladdr("_init_buffers"), 0);
    alwait ();
    alcall ( aladdr("_acquire_and_display"),2, (long) NUMPICS, (long) numpts);
    alwait ();

    fp = fopen ("points", "w");
    for (i=0; i<NUMPICS; i++)
    {
    for (j=0; j<numpts; j++)
       fprintf(fp, "%u\t", 0xff & algetb(VtoP(points+i*numpts+j)));
    fprintf(fp, "\n");
    }
    fclose (fp);

    !_setvideomode ( _DEFAULTMODE );
}

float timestamp (void)
/* return the current time (in seconds) since the counter was last reset */
{
    alcall ( aladdr("_timestamp"), 0);
    alwait ();
    return ( algetfresult() );
}
```

```
ADDR i860malloc (long n)
{
    alcall (aladdr ("_umalloc"), 1, n);
    alwait ();
    return ( algetiresult() );
}
```

```
/*****************************************************************************
|
|  ccd.c
|
|  This program contains the routines for the Hamamatsu - Alacron camera
|  system.  It is controlled from the program ccdctrl.c which runs on
|  the host 486.
|
|  See the file readme.ccd for setup information.
|
|  The "original" directory contains the original files which came with
|  system.
|
|  Author: Jonathan Claman
|
|*****************************************************************************/
#define _HRGDRIVER_

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <i860lib.h>
#include <sys/al860.h>
#include <di.h>
#include <hrg.h>
#include <vlib.h>

#define DIDEV           0
#define HRGDEV          0
#define DI_MODE         3       /* DI acquisition mode */
#define EXPOSURE_TIME   20      /* in milliseconds */

#define BIG_X           1040    /* horizontal resolution of image */
#define BIG_Y           1024    /* vertical resolution of image */
#define IMAGE_SIZE      BIG_X * BIG_Y
#define SMALL_X         520     /* horizontal res. of downsampled image */
#define SMALL_Y         512     /* vertical res. of downsampled image */
#define SMALL_IMAGE     SMALL_X * SMALL_Y
#define PACKED_SIZE     2 + 5*IMAGE_SIZE/4
#define NPICS           40
#define FFTLEN          1024
#define MAXPTS          100

typedef unsigned char pixel8;
typedef unsigned char pixel10;
typedef unsigned short pixel16;

hrguser_t user;         /* global variable containing HRG system info */
```

```
pixel8 *display;      /* global variable pointing to display ram (user.base) */
pixel8 *ram[NPICS+2]; /* global var pointing to storage ram for images */

int num_anal_pts=0;   /* global var containing the number of analysis points */
struct anal_point {   /* global structure of analysis points */
   int x;
   int y;
   pixel8 value[500];
} anal_point[MAXPTS];

extern void *umalloc (int);
void init_buffers (void);
void acquire_and_display (int, int);
void acquire_and_display_once (pixel8 *);
void display_from_memory (void);
void display_from_memory_once (pixel8 *);
void wait_for_data (void);
void take_picture (void);
void dcl_init (void);
void pause (int);
float timestamp (void);
void write_images (int, int);
void read_images (int, int);
void packbits (pixel16 *, pixel10 *);
void unpackbits (pixel10 *, pixel16 *);
int etchrate (pixel16 *, int);
void zeropad (float *, int, int);
void shrink_image (pixel8 *, pixel8 *);
void display_small_image (pixel8 *);
void display_small_windowed_image (pixel8 *, int, int);
void mouse_handler (int, int, int, int, int, int);

/*****************************************************************************
 * The main() function doesn't play a part in slave mode applications, but
 * the compiler likes to see that a main() exists, regardless of what's in it.
 * main() can be executed by running rt860 on this program.
 */
void main (void)
{
   printf("\nAlacron sucks!\n");
}

/*****************************************************************************
 * Allocate memory space for the images.
 */
void init_buffers (void)
{
```

```
    int i;

    for (i=0; i<NPICS; i++)
        ram[i] = umalloc (IMAGE_SIZE);
}

/*************************************************************************
 * Poll the DI to wait for the DMA transfer to complete.
 * The display is automatic as the DMA destination is the display buffer.
 * Copy the data from the display buffer to memory for later use.
 * Save certain pixels in separate arrays if desired.
 * Loop this process for NPICS times.
 */
void acquire_and_display (int pics, int numpts)
{
    pixel8 *r[MAXPTS];
    distatus_t status;
    int i, j;
    float time1, time2, pictime[500];
    FILE *fp;

    display = user.base;
    for (i=0; i<numpts; i++)
    {
        r[i] = display + BIG_X*anal_point[i].y + anal_point[i].x;
    }

    for (i=0; i<pics; )
    {
        diioctl (DIDEV, DI_GET_STATUS_PORT, &status);
        if (!status.START & !status._EF)
        {
            pictime[i] = timestamp();
            take_picture();
            memcpy (ram[0], display, IMAGE_SIZE); /* store image in ram */
            for (j=0; j<numpts; j++)
            {
                anal_point[j].value[i] = *r[j]; /* get points for analysis */
            }
            i++;
        }
    }
    if ((fp = fopen("points", "w")) == NULL)
    {
        printf("Error opening points file\n");
        exit(0);
    }
    for (i=0; i<pics; i++)
```

```
    {
        fprintf (fp, "%.2f\t", pictime[i]-pictime[0]);
        for (j=0; j<numpts; j++)
            fprintf (fp, "%u\t", 0xff & anal_point[j].value[i]);
        fprintf(fp, "\n");
    }
    fclose(fp);
}

void acquire_and_display_once (pixel8 *buf)
{
    distatus_t status;
    int i, j;

    display = user.base;

    for (i=0; i<1; )
    {
      diioctl (DIDEV, DI_GET_STATUS_PORT, &status);
      if (!status.START & !status._EF)
      {
          i++;
          take_picture();
          memcpy (buf, display, IMAGE_SIZE); /* store image in ram */
      }
    }
}

/*************************************************************************
 * Read images from memory and display them on the HRG in an infinite loop,
 * Time the process for the first loop.
 */
void display_from_memory (void)
{
    int i;
    float time1, time2;

    time1 = timestamp();
    for (i=1; i<NPICS; i++) memcpy(display, ram[i], IMAGE_SIZE);
    time2 = timestamp();
    printf("Playback: %.2f fps\n", (NPICS-1)/(time2-time1));
    while (1) for (i=1; i<NPICS; i++) memcpy(display, ram[i], IMAGE_SIZE);
}

void display_from_memory_once (pixel8 *buf)
{
    memcpy(display, buf, IMAGE_SIZE);
}
```

```
/*************************************************************************
 * Wait for the DMA to complete.
 */
void wait_for_data (void)
{
   distatus_t status;

   do
      diioctl (DIDEV, DI_GET_STATUS_PORT, &status);
   while
      (status.START | status._EF);
}

/*************************************************************************
 * A picture is taken by openning the shutter, pausing, and then closing
 * the shutter.  This function also takes care of starting the DMA transfer.
 */
void take_picture (void)
{
   int i;

   diioctl (DIDEV, DI_SET_DOUTPUT, 0);  /* open the shutter */
   pause (EXPOSURE_TIME);
   diioctl (DIDEV, DI_START);           /* prepare the DMA */
   diioctl (DIDEV, DI_SET_DOUTPUT, 2);  /* close the shutter */
}

/*************************************************************************
 * Perform the initialization of the real time clock, the HRG, and the DI.
 */
void dcl_init (void)
{
   unsigned char crtcval[N_CRTC_REGS] = { /* 1040x1000 display resolution */
      80,            /* r0 Nht */
      65,            /* r1 Nhd */
      71,            /* r2 Nhsp */
      0x2b,          /* r3 Nvsw,Nhsw */
      127,           /* r4 Nvt */
      0,             /* r5 Nadj */
      125,           /* r6 Nvd */
      126,           /* r7 Nvsp */
      0,             /* r8 mode */
      7,             /* r9 Nr */
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      1,             /* r27 Nvad */
      0,0,0,0,0,0,0,0,0,0,0,0};
```

```
    rtcioctl(RTC_SETFREQ, 1000);

    hrgerrorprt (1);
    if (hrgopen (HRGDEV))
    {
        printf ("ERROR: open HRG device %d failed\n", HRGDEV);
        exit (1);
    }
//    printf ("HRG Opened ok\n");

    hrgioctl (HRGDEV, HRG_SET_RESOLUTION, R1024x768);
    hrgioctl (HRGDEV, HRG_SET_CRTC, crtcval);
    hrgioctl (HRGDEV, HRG_GET_USER_PARAMS, &user);
    hrgioctl (HRGDEV, HRG_SET_DEFAULT_PALETTE);
    memset (user.base, 0, BIG_X*BIG_Y); /* clear the screen */
    hrgioctl (HRGDEV, HRG_DISABLE_PASSTHROUGH);

    dierrorprt (1);
    if (diopen (DIDEV) ≠ 0)
    {
        printf ("ERROR: open DI device %d failed\n", DIDEV);
        exit (1);
    }
//    printf ("DI Opened ok\n");
    diioctl (DIDEV, DI_SET_DOUTPUT, 2);  /* close the shutter */
    diioctl (DIDEV, DI_STOP);
    diioctl (DIDEV, DI_SET_TRANSCOUNT, 0);
    diioctl (DIDEV, DI_SET_IF_CONTROL, 1);
    diioctl (DIDEV, DI_SET_IF_DATA, 1);
    diioctl (DIDEV, DI_SET_WORDSIZE, 8);
    diioctl (DIDEV, DI_ENABLE_DMA);
    diioctl (DIDEV, DI_SET_INPUT_MODE, DI_MODE);
    diioctl (DIDEV, DI_SET_DMA_ADDR, user.base);
}

/*************************************************************************
 * Pause for time, in milliseconds.
 * Don't reset the counter to be nice to other users of the clock.
 * Assume that the RTC_FREQ is set to 1000 Hz!!!
 */
void pause(int time)
{
    long int time1, time2;

    rtcioctl(RTC_GETCOUNT, &time1);
    do rtcioctl(RTC_GETCOUNT, &time2);
    while (time2-time1 < time);
}
```

```
/*************************************************************************
 * Return the current time (in seconds) since the counter was last reset.
 * Assume that the RTC_FREQ is set to 1000 Hz!!!
 */
float timestamp (void)
{
   long int time;
   rtcioctl(RTC_GETCOUNT, &time);
   return(time/1000.0);
}

/*************************************************************************
 * This function takes a starting number and an ending number, and writes
 * images to disk with the filename "d:\data\image##".
 */
void write_images (int startnum, int stopnum)
{
   FILE *fp;
   char *filename;
   int i;
   float time1, time2;

   time1 = timestamp();
   for (i=startnum; i<stopnum+1; i++)
   {
      sprintf(filename, "d:\\data\\image%d", i);
      if((fp=fopen(filename, "wb")) == NULL) return;
      fwrite(ram[i], IMAGE_SIZE, 1, fp);
      fclose(fp);
   }
   time2 = timestamp();
   printf("Write to disk: %.2f fps\n", (startnum-stopnum+1)/(time2-time1));
}

/*************************************************************************
 * This function takes a starting number and an ending number, and reads
 * images from disk with the filename "d:\data\image##".
 */
void read_images (int startnum, int stopnum)
{
   FILE *fp;
   char *filename;
   int i;
   float time1, time2;

   /* this assumes ram[i] has been umalloc'ed already using init_buffers() */
```

```
    time1 = timestamp();
    for (i=startnum; i<stopnum+1; i++)
    {
        sprintf(filename, "d:\\data\\image%d", i);
        if((fp=fopen(filename, "rb")) == NULL) return;
        fread(ram[i], IMAGE_SIZE, 1, fp);
        fclose(fp);
    }
    time2 = timestamp();
    printf("Read from disk: %.2f fps\n", (startnum-stopnum+1)/(time2-time1));
}

/*************************************************************************
 * This function takes in a pointer to an image that has 10 bit values
 * aligned to 16 bit chunks. It packs the data into a new location,
 * returns that new location, and frees the original space.
 */
void packbits(pixel16 *input, pixel10 *output)
{
    int i, j;

    for (i=0, j=0; i<IMAGE_SIZE; i+=4, j+=5)
    {
        output[j]   =   input[i]   & 0x00ff;
        output[j+1] = ((input[i]   & 0x0300)>>8) + ((input[i+1] & 0x003f)<<2);
        output[j+2] = ((input[i+1] & 0x03c0)>>6) + ((input[i+2] & 0x000f)<<4);
        output[j+3] = ((input[i+2] & 0x03f0)>>4) + ((input[i+3] & 0x0003)<<6);
        output[j+4] = ((input[i+3] & 0x03fc)>>2);
    }
}

/*************************************************************************
 * This function is the reverse of bitpacker. It takes in a pointer
 * to an image that has 10 bit values packed in 8 bit chunks. It unpacks
 * the data into a new location, returns that new location, and frees the
 * original space.
 */
void unpackbits(pixel10 *input, pixel16 *output)
{
    int i, j;

    for (i=0, j=0; i<IMAGE_SIZE; i+=4, j+=5)
    {
        output[i]   = ((input[j]   & 0xff)>>0) + ((input[j+1] & 0x03)<<8);
        output[i+1] = ((input[j+1] & 0xfc)>>2) + ((input[j+2] & 0x0f)<<6);
        output[i+2] = ((input[j+2] & 0xf0)>>4) + ((input[j+3] & 0x3f)<<4);
        output[i+3] = ((input[j+3] & 0xc0)>>6) + ((input[j+4] & 0xff)<<2);
    }
```

```
}

/*************************************************************************
 * This function computes the etchrate in Angstroms per minute.
 * Inputs: pointer to a vector containing pixel intensities over time,
 *         and an integer containing the length of this vector.
 * Output: int containing the etchrate.
 */
int etchrate(pixel16 *data, int datalength)
{

/* this stuff needs a lot of help!!! */

    float mean;
    float hammeddata[FFTLEN];
    float nobias[1024];
    COMPLEX fftresult[1024]; // remember that bldwts() necessary if > 1024
    COMPLEX sqmagfft[512];
    float maxvalue, maxindex;
    int lambda = 7534, sr = .9, n_poly = 3.73; // sr should be calculated!

    hamm (data, 1, hammeddata, 1, FFTLEN);
    meanv (hammeddata, 1, &mean, FFTLEN);
    mean = -mean;
    vsadd (hammeddata, 1, &mean, nobias, 1, FFTLEN);
    zeropad (nobias, datalength, 1024); // need to write this function
    rfftb (nobias, fftresult, 1024, 1);
    rfftsc (fftresult, 1024, 2, 0);
    cvmags (fftresult, 1, sqmagfft, 1, 512);
    maxv (sqmagfft, 1, &maxvalue, &maxindex, 512);

    return ((int) (maxindex * lambda * sr * 60)/(1024 * 2 * n_poly));
}

void zeropad (float *input, int length1, int length2)
{
    int i;

    for (i=length1; i<length2; i++)
        input[i] = 0;
}

/*************************************************************************
 * This function downsamples an image.  Is there a faster, more efficient
 * way to do this, maybe using vmov() with a stride of xinc?
 */
void shrink_image(pixel8 *small, pixel8 *big)
{
```

```
    int i, j, k=0, xinc, yinc;

    yinc = BIG_Y / SMALL_Y;
    xinc = BIG_X / SMALL_X;

    for (i=0; i<BIG_Y; i+=yinc)
        for (j=0; j<BIG_X; j+=xinc)
            small[k++] = big[i*BIG_X+j];
}

/*****************************************************************************
 * This function displays a shrunk image at full size
 */
void display_small_image(pixel8 *q)
{
    int i, j, k, l, yinc, xinc;
    pixel8 *p, *r;

    yinc = BIG_Y / SMALL_Y;
    xinc = BIG_X / SMALL_X;

    for (i=0; i<SMALL_Y; i++)
        for (j=0; j<SMALL_X; j++)
            for (k=0; k<yinc; k++)
                for (l=0; l<xinc; l++)
                {
                    p = (pixel8 *)user.base + BIG_X*(k+yinc*i) + l+xinc*j;
                    r = q + SMALL_X*i + j;
                    *p = *r;
                }
}
/*****************************************************************************
 * This function displays a shrunk image in a window with coordinates
 * (wx, wy)
 */
void display_small_windowed_image(pixel8 *q, int wx, int wy)
{
    int i,j;
    pixel8 *p, *r;

    for (i=0; i<SMALL_Y; i++)
        for (j=0; j<SMALL_X; j++)
        {
            p = (j+wx) + BIG_X * (i+wy) + (unsigned char *)user.base;
            r = q + SMALL_X * i + j;
            *p = *r;
        }
}
```

```
/*************************************************************************
 * This function takes an (x,y) coordinate and a mouse button (b).
 * It displays a mouse cursor on the HRG, and marks pixels white if
 * the button is pressed.
 */
void mouse_handler (int b, int x, int y, int oldb, int oldx, int oldy)
{
   int i;

   if (x ≠ oldx) /* only if x position has moved */
   {
      for (i=0; i<BIG_Y; i++) // restore old line
         display[BIG_X*i + oldx] ^= 0xff;
      for (i=0; i<BIG_Y; i++) // draw new cursor line
         display[BIG_X*i + x] ^= 0xff;
   }
   if (y ≠ oldy) /* only if y position has moved */
   {
      for (i=0; i<BIG_X; i++)
         display[BIG_X*oldy + i] ^= 0xff;
      for (i=0; i<BIG_X; i++)
         display[BIG_X*y + i] ^= 0xff;
   }
   /* set analysis point of mouse button pressed */
   if (b == 1 && oldb ≠ 1)
   {
      display[BIG_X*y + x] = 255;
      anal_point[num_anal_pts].x = x;
      anal_point[num_anal_pts++].y = y;
   }

/* eventually should have a way to remove points, too */

}
```