UNIVERSITY OF CALIFORNIA, SANTA BARBARA

UNDERGRADUATE THESIS

Photoacoustic Phenomena in Diffusion-Limited Aggregates

Author:
Morgan BRUBAKER

Supervisor:
Dr. David WELD

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Weld Lab
Department of Physics

June 15, 2017

Declaration of Authorship

I, Morgan BRUBAKER, declare that this thesis titled, "Photoacoustic Phenomena in Diffusion-Limited Aggregates" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

University of California, Santa Barbara

Abstract

Dr. David Weld
Department of Physics

Bachelor of Science

Photoacoustic Phenomena in Diffusion-Limited Aggregates

by Morgan BRUBAKER

We have used a thermal evaporation process to produce a class of thin films, known as diffusion-limited aggregate (DLA) films, that have a distinct fractal nanostructure. These films have a number of interesting properties, such as high optical absorption and poor thermal transport. These properties make DLA films ideal for the application of photoacoustics, in which an amplitude-modulated light source heats a material periodically, causing thermal and volume fluctuations in both the absorbing material and the surrounding medium, which generate pressure waves in the medium. A DLA film used in conjunction with a modulated multi-element light source constitutes an optically-addressed acoustic/ultrasonic phased array capable of producing nearly-arbitrary sound fields. The ability to produce arbitrary sound fields is limited by the diffraction limit of sound, which one might hope to overcome by taking advantage of nonlinear phenomena.

Chapter 1 defines diffusion-limited aggregate films and describes the means of their production and their general properties. Chapter 2 discusses the photoacoustic effect and outlines how the properties of diffusion-limited aggregate films affect their performance as photoacoustic transducers. Chapter 3 characterizes the phased array used in our experiments and discusses some details of its operation, and also presents the results of our efforts to generate patterned ultrasound. Chapter 4 explains how the diffraction limit of sound restricts our ability to produce arbitrary sound fields and details our efforts to overcome this limit via nonlinear phenomena. Chapter 5 delineates other directions for future research on these films, both those that have been pursued and those that could perhaps prove fruitful.

Acknowledgements

I would like to thank Dr. Weld for providing me with the opportunity to perform research in his lab and serving as an excellent mentor thoughout my time in his lab, along with the members of the Weld lab for their constant support. I would especially like to thank Krutik Patel and Eli Wolf for their instruction and support in the early stages of my research. I would also like to express my gratitude towards the Worster family for providing funding for my research over the summer of 2016 through the Worster fellowship, along with Kevin Singh for serving as my mentor during the fellowship, and the Hellman Foundation for providing additional funding for the project as a whole. The College of Creative Studies has also earned my thanks, both for providing funding over the summer of 2015 through the SURF program and for continued support throughout my undergraduate career. I would also like to express the deepest gratitude to Dr. Kazimir Gasljevic, who enabled a good portion of this research to take place by allowing me to borrow is sound level meter for quite some time, and Mark Cornish for his patience while guiding me through the process of using a scanning electron microscope.

Contents

List of Figures

List of Tables

Chapter 1

Diffusion Limited Aggregates and their Synthesis

This thesis discusses a class of materials known as diffusion-limited aggregates and their various applications, with special emphasis on photoacoustic applications. Before discussing how these materials are used and studied in this research, we must establish a firm understanding of what these materials are; for this reason, we begin by discussing the process by which these materials are produced and some common properties of these films.

1.1 Diffusion-Limited Aggregates

Diffusion-limited aggregation is a process by which particles randomly coalesce to form a larger aggregate. The process gets its name from the fact that the rate of formation of the aggregate is limited primarily by the rate at which a particle can diffuse through the medium in which aggregation takes place.

The process of diffusion-limited aggregation is perhaps best described by outlining how one would simulate the process. Suppose one initially had a single stationary particle in an otherwise empty region. Then suppose another particle enters this empty region with a random velocity at a random position on the region's boundary. If the moving particle, while traveling with constant velocity, comes close enough to the stationary particle (i.e. if the particles collide), then the two particles has some probability of adhering (usually assumed to be 100%), in which case the particle stops moving. Meanwhile, if the moving particle does not collide with the stationary particle, then it simply exits the region. In either case, a new moving particle is generated, and this particle likewise either adheres to one of the stationary particles or exits the region. This process of randomly sending a particle into the region of interest and seeing if it collides with the stationary particles is repeated until some condition is met - for instance, a sufficient number of particles have come to rest by colliding with the aggregate - and the body formed by the particles that have stuck together is referred to as a diffusion-liimited aggregate [1].

From this description, the key aspects of the process of diffusion-limited aggregation become clear. In order for diffusion-limited aggregation to occur, the paths of the constituent particles must somehow be randomized during the formation process, and the particles must be able to adhere to each other following a collision.

1.2 Synthesis

The process by which diffusion-limited aggregate (DLA) films are produced bears strong resemblance to a typical thermal evaporation. A small amount of the material to be evaporated is placed in a boat or basket inside a vacuum chamber. Once a sufficiently low pressure is achieved in the vacuum chamber, a current is run through the boat or basket, which

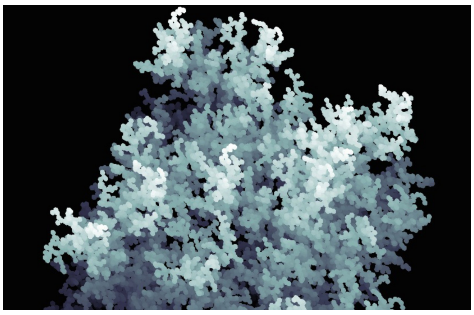

FIGURE 1.1: Simulated DLA structure. Results are from a simulation by Alexander "Shura" Kotlerman.

resistively heats the container and the material. This causes some particles of the material to evaporate, and these particles are eventually deposited on a nearby substrate.

The main difference between our evaporation process and a typical thermal evaporation is the pressure of the vacuum chamber during the evaporation. Under normal circumstances, one would hope to perform thermal evaporation at as low of a pressure as possible, typically on the order of 10^{-6} Torr. One reason for this low pressure is that this maximizes the distance particles can travel without undergoing a collision, a parameter known as the mean free path. The mean free path of gaseous atoms is given by[2]

$$l = \frac{1}{\pi d^2 n} \tag{1.1}$$

where d is the maximum distance betwwen atoms at which a collision occurs and n is the number density of the gas. Using the ideal gas law, we have

$$n = \frac{N}{V} = \frac{p}{k_B T} \tag{1.2}$$

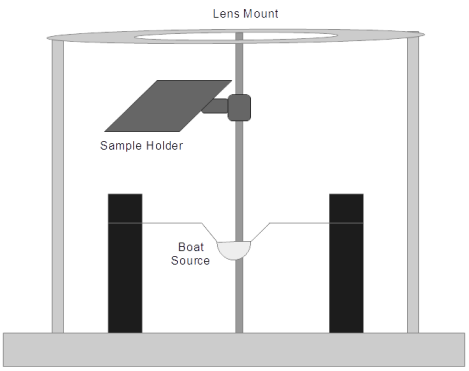

FIGURE 1.2: A diagram of the vacuum chamber. The material to be evaporated is placed in the boat or basket suspended between two large metal posts. The substrate is placed somewhere in the chamber; large substrates can be placed overhead on the annular "Lens Mount", while smaller samples can be placed on the bottom of the chamber, suspended in an overhead mount such as the slide holder shown (not currently attached to the evaporator), or otherwise placed in a position where deposition is expected.

where p is the pressure, k_B is the Boltzmann constant, and T is the temperature. Substitution for n gives the result

$$l = \frac{k_B T}{\pi p d^2} \tag{1.3}$$

From this equation, we see that minimizing the pressure maximizes the mean free path, so evaporating at low pressures allows for more fine control of film deposition. For instance, for a bismuth atom in argon, assuming that the diameter d is the sum of the Van der Waals radii of a bismuth atom and an argon atom, the typical pressure of 10^{-6} Torr yields a mean free path of 8.4 km at room temperature, and this value only increases as the temperature rises; since this is much smaller than the size of a typical evaporation chamber, one can typically assume that an evaporating particle will travel in a straight line from the source to the substrate. Our process, on the other hand, requires much higher pressures, typically on the order of 2 Torr. This comparatively high pressure limits the mean free path of the evaporated particles, meaning the particles undergo a random walk; the high pressure thus provides the randomization necessary for diffusion-limited aggregation to occur. Quantitatively, this pressure corresponds to a mean free path of 4.2mm at room temperature, or 2.6 cm at the (atmospheric pressure) boiling point of bismuth; since this mean free path is much smaller than the dimensions of our evaporation chamber, the evaporated bismuth particles will undergo a random walk in the air. During this random walk, the particles will undergo many collisions with other evaporating particles. When evaporated particles collide with each other, they have a chance to stick together, forming small clusters; each of these clusters eventually collides with a surface in the evaporator, and often enough this surface will be the intended substrate. Note that as the clusters increase in size, the diameter of the cluster increases and the mean free path decreases, causing more frequent collisions.

The general process as outlined above successfully produces DLA films for virtually every material that we have attempted to evaporate; however, parameters such as background pressure, current, duration, and evaporative source can affect the deposition results, and the recommended values of these parameters vary depending on the material being evaporated. For specific details on how to prepare films of various materials, please reference Appendix A, which contains notes from individual evaporations which can be used as a reference; for the sake of elaborating on some details of the evaporation process, we will outline in detail how to perform an evaporation of bismuth.

1.2.1 More Detailed Process

The process of preparing a DLA film consists of three main steps: preparing the evaporant and substrate in the chamber, sealing the chamber, and performing an evaporation.

Before the evaporation begins, both the material to be evaporated and the intended substrate must be placed in the chamber. The evaporant should be placed in a boat or basket which is suspended between two metal posts in the chamber; the sources typically used are a molybdenum boat and a tungsten basket, depending on the material. For a bismuth evaporation, place 3-4 small pellets of bismuth in a molybdenum boat; less bismuth may be required if there is a sufficient amount of bismuth remaining from a previous evaporation.

There are a number of options for placing the intended substrate within the chamber. Most small substrates, such as copper films or glass slides, can either be hung in a sample holder mounted to a vertical post in the chamber, or simply placed on the bottom of the chamber, either flat or propped up against something to achieve an angle. Since the evaporation produces particles and aggregates with randomized velocities, some deposition onto the substrate should be expected regardless of the substrate's position, though for a good deposition, the position of the substrate should be chosen such that there is an uninterrupted line-of-sight between it and the evaporative source, and orienting the substrate such

that it is normal to this line of sight may help yield a more uniform deposition. Larger substrates, such as the large mirror blank introduced in Chapter 4, can be placed on the annular platform at the top of the chamber, such that the area where deposition is desired is placed over the hole in the platform's center. The large distance from the boat to this platform helps to increase uniformity of deposition, but also decreases the deposition rate compared to deposition for a closer target under the same conditions; a longer evaporation with more material and/or a lower pressure may be required in order to make up for this reduced deposition rate. Regardless of the substrate's position, the chamber should be cleaned with isopropanol and KimWipes before proceeding.

Once the evaporant and substrate are in place, the vacuum chamber is ready to be sealed. First, a cylindrical shield is placed around the chamber; this shield blocks evaporated particles from depositing onto the walls of the vacuum chamber. There is a small hole in this shield through which one could look into the chamber during an evaporation; the shield should be oriented such that one can easily look through this hole. This detail is more important for evaporations of other materials where, unlike bismuth, the temperatures required are large enough that the evaporative source begins to glow, allowing one to easily see inside the chamber.

A bell jar is then placed over the system, encapsulating the shield. The bell jar is critical to achieving vacuum, as it is the seal between the O-ring on the rim of the jar and the base of the evaporator, along with that between the O-ring and the bell jar, that holds vacuum. To ensure a tight seal, the base of the evaporator and the O-ring should be cleaned with isopropanol and KimWipes, and Dow Corning high vacuum grease should be applied to the O-ring, before placing the bell jar. Once a tight seal has been established, a cylindrical wire mesh is then placed over the bell jar; this mesh blocks large pieces of glass or other debris in the event that the bell jar shatters or vacuum is otherwise broken.

After ensuring that the chamber is fully sealed (e.g. checking that all O-ring connections are in place; typically at least one connection must be reestablished, since vacuum is usually broken after an evaporation by undoing one of these connections, allowing the sample to be retrieved), the vacuum pump, located in the core, is turned on. The evaporation is typically performed in about 2 Torr of argon; prior to the evaporation we repeatedly establish a significantly lower pressure (roughly in the range 25-55 mTorr) and flush the chamber with argon in order to remove atmospheric air from the chamber. During this process and evaporation, the pressure in the chamber is measured with a pressure meter. A large valve can open or close the connection between the chamber and the vacuum pump, and a smaller valve nearby allows argon to flow into the chamber (so long as there is argon in the connected tank and this tank's valve is also open); the process of flushing the vacuum consists of closing the connection between the chamber and pump, filling the chamber to 200 Torr of argon, waiting a few seconds to allow the argon to mix with the atmospheric air remaining in the chamber, then opening the connection to the pump. Typically, four such flushes are performed.

Once the vacuum has been sufficiently flushed, the chamber is filled with about 2 Torr of argon. One may either fill the chamber with this much argon and close the connection to the pump, or leave the connection to the pump open and constantly flow argon through the chamber to maintain constant pressure. While both options have their theoretical benefits - a constant argon flow can help maintain low oxygen concentrations even in the presence of a leak, while having no air flow should lead to more isotropic and uniform deposition - there is usually not a significant demonstrable difference in the quality of produced films. A timer and a multimeter capable of measuring current should be procured before the evaporation is started; the latter should have a ring clamp such that the current measured is that which flows through the ring, and should be clamped around a large wire just below the base of the evaporator.

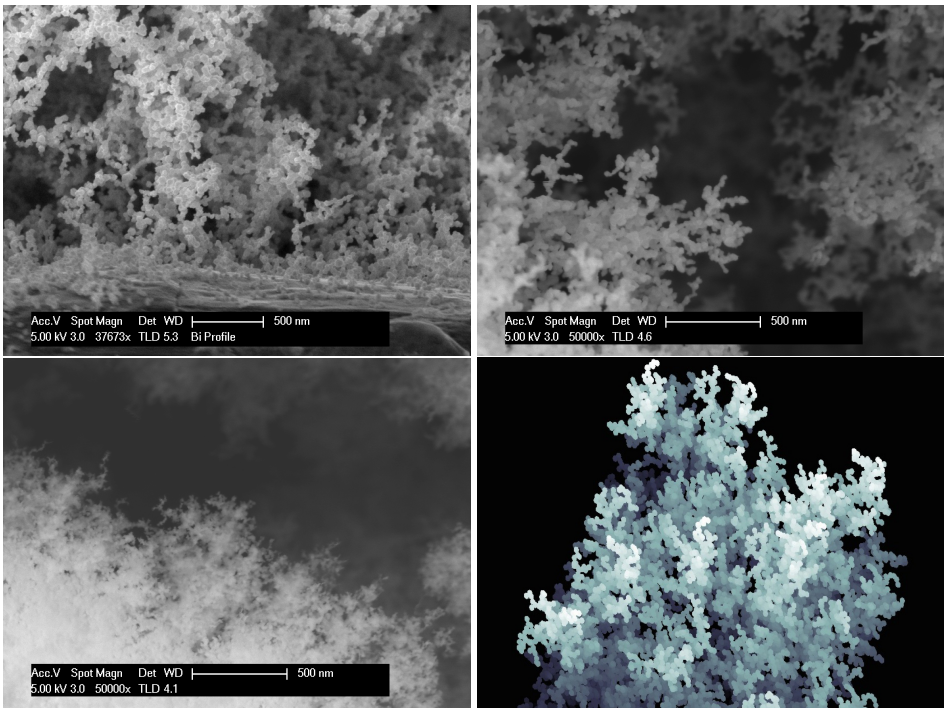

FIGURE 1.3: Scanning Electron Micrographs of DLA films of various materials. Top left: Bismuth. Top right: Copper. Bottom left: Aluminum (SEM taken by Shura). Bottom right: The simulated result shown in Figure 1.1, replicated here for comparison.

The evaporator can be turned on by flipping the rightmost of three switches and then **slowly** turning the large black dial above it. This dial controls the current through the boat, which should be increased by 20 A every 30 seconds until a current of 115 A is reached; this slow increase in current helps prevent damage to the evaporative source. Once the current reaches 115 A, the evaporation is allowed to proceed for another 4-5 minutes. Once the 4-5 minutes have passed, the current is slowly reduced to 0. At this point all that remains is to allow the sample to cool; to this end, the connection to the vacuum pump is closed, the chamber is filled with 200 Torr of argon, the vacuum pump is turned off, and the sample is allowed to sit for around an hour. After sufficient cooling time has passed, the chamber is depressurized by undoing an O-ring connection then opening the valve controlling the connection to the pump, and the sample is retrieved.

1.3 Properties

The films produced by this method typically possess a number of interesting properties. These properties are as follows:

1) These films typically have a particular nanostructure that can be described as treelike, consisting of branches of small grains. See Figure 1.3 for scanning electron microscope (SEM) images of a selection of these films. The strong qualitative resemblance between these structures and the structure of a simulated diffusion-limited aggregate suggests that this structure is a direct consequence of the process of diffusion-limited aggregation. It should be noted, however, that there are some DLA films that exhibit slightly different structures, such as iron and nickel (See Figure 1.4) The cause of this altered structure is not known and has not been thoroughly investigated, although we suspect that the magnetic properties of iron and nickel play a role in altering the structure.

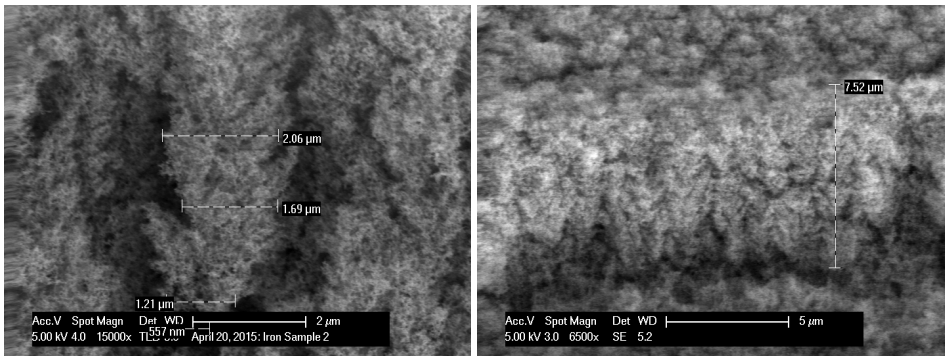

FIGURE 1.4: Scanning Electron Micrographs of DLA films of iron (left) and nickel (right). Note that the structure does not quite resemble the results that we'd expect for a DLA film, as the structure appears much more compact and linear. The magnetic properties of both iron and nickel are the suspected cause of the different structure, although this has not been proven.

2) The films appear black, meaning that they absorb most of the light incident upon them over a wide bandwidth. Figure 1.5 shows results of a measurement of the reflectivity of a copper DLA film; reflectivity of less than 2% is observed for nearly the entire measured spectrum. This means that the remaining 98% or more of the light is either transmitted or absorbed; the fraction of light that is transmitted rather than absorbed seems to primarily be a function of the film's thickness. We have not thoroughly investigated why this nanostructure results in strong broadband optical absorption. One might predict broadband absorption by observing that an absorbed wavelength typically corresponds to a characteristic length in the material's structure, and that the randomized structure of a diffusion-limited aggregate means that there is not a single characteristic length, but rather a range of characteristic wavelengths, causing the film to absorb at all wavelengths within this range. However, this is only one possible explanation for a poorly-understood phenomena, and further research into this topic would be justified.

3) The films have poor thermal conductivity; heat generated in one area of the film

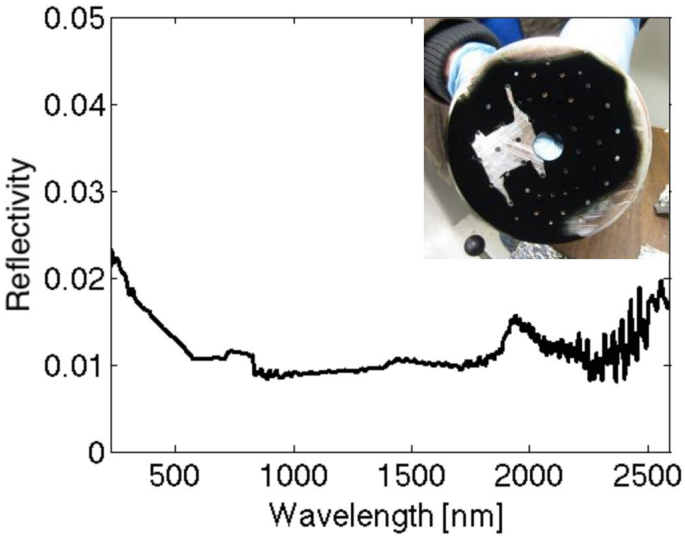

FIGURE 1.5: Reflectivity data for a copper DLA film; note that the reflectivity is around 1% over a wide bandwidth, and stays below 2% for all but the shortest of measured wavelengths.

does not spread quickly to the rest of the film. This could be seen as a consequence of the structure of the films, since poor mechanical connections between different portions of the film suggest poor thermal transport. For a more thorough investigation of the thermal transport properties of some of these films, see Eli Wolf's thesis [3].

These properties appear rather consistently, and do not seem to depend sensitively on parameters such as the evaporation pressure, temperature or time, or the material used to produce the film. This lack of sensitivity suggests that the formation of DLA films depends on classical mechanics rather than chemistry. To date, we have successfully deposited DLA films of silver, gold, copper, iron, tin, nickel, bismuth, aluminum, germanium, indium and silicon monoxide, and while there is some variance between materials - for instance, bismuth seems to perform the best in photoacoustic applications - that a process characterized by randomness would reliably produce materials with such consistent properties is somewhat remarkable.

Chapter 2

Photoacoustic Effect in DLA Films

Having discussed the formation and properties of diffusion-limited aggregate films in the previous chapter, we now look to discuss the main application of these films as pursued in this research: the production of acoustic signals via the photoacoustic effect. However, in order to properly discuss this application, we must first understand the physical process by which these films are able to generate sound. Thus, in this chapter we will explain the photoacoustic effect in general, and elaborate why DLA films are capable of producing a strong photoacoustic signal.

2.1 The Photoacoustic Effect

The photoacoustic effect is a process by which an amplitude-modulated light source causes a material to generate sound at the frequency of amplitude modulation. The photoacoustic effect depends on the phenomenon of thermal expansion, in which the density of a material decreases when its temperature is increased. When a material absorbs light, it gains the light's energy as heat . If the system was in thermal equilibrium before a constant light source is turned on, then the light will cause the material's termperature to increase. This will cause the material to expand, and if the heat is able to spread quickly to other portions of the material or the surrounding medium, then these will also undergo thermal expansion; for instance, if the material is in air, then the nearby air will also undergo expansion.

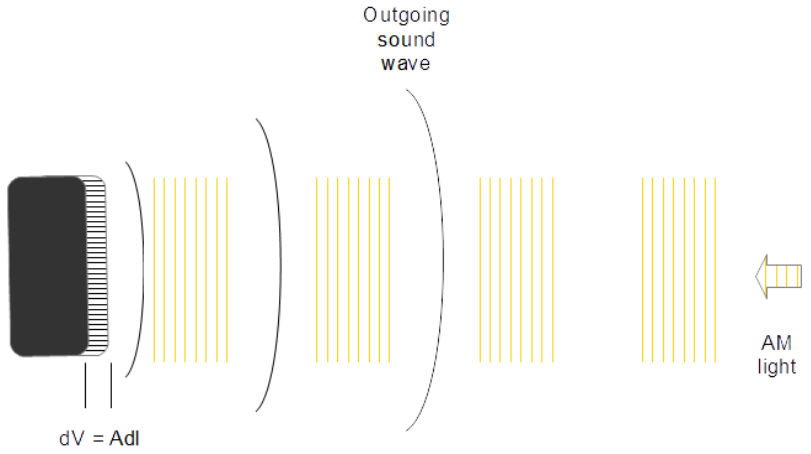

FIGURE 2.1: A cartoon illustrating the photoacoustic effect. Amplitude-modulated (AM) light is incident upon a material, which absorbs the light. The heat gained by the material causes a volume fluctuation in the material (and the surrounding medium, and anything to which the heat can spread quickly), which in turn generates a pressure wave, or sound, in the surrounding medium.

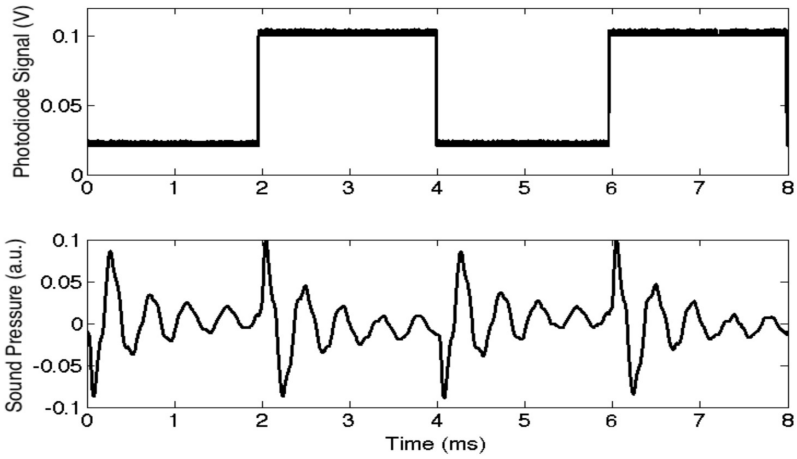

FIGURE 2.2: A plot demonstrating the photoacoustic response of a DLA film. A DLA film is exposed to a square wave of light (top plot), and the resulting acoustic signal has positive pressure pulses when the light turns on and negative pressure pulses when the light turns off. The further oscillations between the light turning on and the light turning off is believed to be due to ringing in the microphone.

Conversely, if one turns off a light source after the material has been able to heat up due to the light's presence, the material and any thermally-connected matter will cool and contract.

Now suppose one has a light source that is turning on and off at a given frequency. So long as the frequency is not so great that the material does not have time to change its temperature, the material and any thermally-connected matter will undergo a temperature oscillation, and thus a density oscillation, at the modulation frequency. The density oscillation within the material and any thermally-connected matter drives a pressure wave, or a sound wave, in the surrounding medium, and this sound wave will have the same frequency as the temperature oscillations and hence the light modulation.

2.2 DLA Films as Ideal Photoacoustic Transducers

One may note that the description of the photoacoustic effect given above applies quite generally to a wide range of materials; there are only three requirements for a material to be capable of generating a photoacoustic signal:

- The material must be capable of absorbing light at some frequency, and the light source used must emit light at that frequency.
- The material must undergo thermal expansion; if the material itself does not undergo thermal expansion, a photoacoustic signal can still be generated if the material is thermally connected to something that does undergo thermal expansion. A surrounding medium such as air is typically sufficiently thermally connected to the material and capable of thermal expansion, so this requirement is not a strong restriction.
- There must be a surrounding medium for the generated pressure wave to travel in.

These restrictions do not greatly constrain the choice of material for generating photoacoustic sound waves, and in principle one expects that practically any material is capable of generating a photoacoustic signal. While this is true, the amplitude of the photoacoustic signal produced by most materials is weak enough that an audible signal would require a prohibitively strong light source to produce. However, DLA films exhibit a significantly

stronger photoacoustic response than many other materials. For instance, we have determined by direct measurement that the photoacoustic response at 40 kHz of a bismuth film is over an order of magnitude stronger than that of a block of graphite. Less quantitatively, the response is strong enough that, if one were to illuminate a DLA film with a 1-LED flashlight on a low brightness setting (such that the lower brightness is achieved by amplitude-modulation at a frequency between 20 Hz and 20 kHz), one would be able to detect the signal produced by ear. The strong photoacoustic response of these films can be seen as a direct consequence of the properties discussed in Chapter 1. The strong optical absorption means that these films are relatively efficient when converting incoming light into heat, and the poor thermal conductivity means that this heat will be more or less confined to the portion of the film that absorbed it, and will only spread into the nearby air by conduction. This allows for the films to establish larger local thermal fluctuations, which results in larger and more localized heat transfer to the nearby air, larger volume fluctuations in the air (and the material, although we assume that the air's expansion is the dominant effect), and thus a stronger signal. These same properties cause these films to have a small thermal response time constant, allowing them to switch temperatures rapidly and thus generate sound at high frequencies.

To demonstrate that the properties of the films described above do, in fact, increase the photoacoustic signal, suppose that a pulse of light containing N photons is incident upon an arbitrary material. Assume that the material absorbs a fraction A of the N photons incident upon it; then the heat gained by the material is given by $AN\hbar\omega$. To find the change in termperature corresponding to this heat gain, we divide by the heat capacity of the material:

$$\Delta T = \frac{A}{C} N\hbar\omega \tag{2.1}$$

Suppose that we generate sound via the photoacoustic effect using such pulses of light, with the time between pulses being such that the material is able to cool to its original temperature in the time between pulses; then ΔT is the amplitude of temperature oscillations in the material. The rate of conductive heat transfer to the nearby air and the thermal expansion and contraction that produces the photoacoustic signal are both roughly proportional to the amplitude of the temperature oscillations, so to maximize the photoacoustic response for a given light source, we must maximize ΔT; this is equivalent to maximizing the fraction $\frac{A}{C}$, so the photoacoustic effect is strongest when the absorption coefficient A is large and the heat capacity C is small. Because DLA films have strong optical absorption, A is large for a DLA film. Moreover, the poor thermal transport properties of DLA films means that we can regard the region that absorbs the photons as effectively thermally isolated from the rest of the film; therefore, the relevant heat capacity is not that of the film as a whole, but that of the region absorbing the photons, which is considerably smaller than that of the whole film due to its reduced size. Thus, the fraction $\frac{A}{C}$ is quite large for a DLA film, which implies a strong photoacoustic response.

By similar arguments, DLA films in air can be shown to have low thermal relaxation times, meaning that they are capable of changing their temperature very quickly. We assume that the rate at which a grain of the DLA film loses heat is just the rate at which heat is lost to the surrounding air by conduction, and that any other means of heat dissipation are negligible. Then the rate at which energy is lost by diffusion of heat into the nearby air is given by

$$C\frac{dT}{dt} = \alpha\Gamma k_B \Delta T \tag{2.2}$$

where $\alpha \approx 0.15$ is a numberical parameter that describes the degree to which an air particle is thermalized after colliding with the grain once, ΔT is the temperature difference between

the grain and the surrounding gas, and Γ is the collision rate of gas particles with the grain. Γ can be expressed as nva, where n is the number density of the gas, v is the RMS velocity of the gas particles, and a is the cross-sectional area of the grain. Noting that $\frac{dT}{dt} = \frac{d\Delta T}{dt}$, one has the differential equation

$$\frac{d\Delta T}{dt} = -\frac{\alpha \Gamma k_B}{C} \Delta T \tag{2.3}$$

The solution for ΔT is a decaying exponential $Ae^{\frac{-t}{\tau}}$ with $\frac{1}{\tau} = \frac{\alpha \Gamma k_B}{C}$; this inverse time can be viewed as an upper bound on the frequencies at which the films can change their temperatures. As with the signal strength, the inverse dependence on C means that the poor thermal conductivity of the films leads to a high upper bound on frequency. If we assume that the photons are absorbed by a single grain of the film, such that the relevant heat capacity C is that of a single grain, then a bismuth film with 10nm grains has a maximum frequency of about 1 MHz. This maximum frequency is well above the upper threshold for the audible range of 20 kHz, meaning that DLA films are capable of producing ultrasound, or sound with frequencies above the threshold for human hearing. Thus, DLA films are able to function as broadband transducers for both acoustic and ultrasonic applications. For more detailed calculations regarding the form of the sound produced by a DLA film, see Krutik's thesis [4].

2.2.1 Possible Limitations

Maximum Optical Intensity/Damage Threshold

Although the structure of DLA films allows them to produce strong photoacoustic signals, this structure also makes the films susceptible to damage through use. When subjected to regular illumination in our experimental apparatus (to be discussed in Chapter 3), DLA films begin to show visible signs of damage on timescales ranging from a few days to two weeks. Portions of the film which are not subject to regular illumination appear undamaged, while regions that are exposed to regular illumination turn to a somewhat translucent

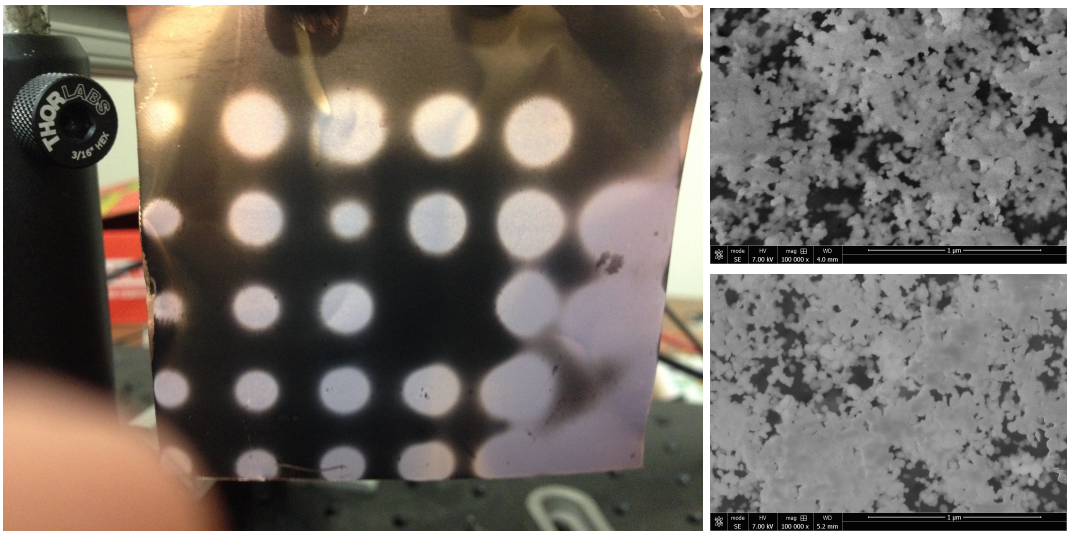

FIGURE 2.3: Left: A copper DLA film deposited on copper. The film was exposed to amplitude-modulated light from a grid of LEDs for an extended period of time; the areas of the film exposed to this light were damaged, turning white. Right: SEM images of an undamaged portion of the film (top) and a damaged portion of the film (bottom).

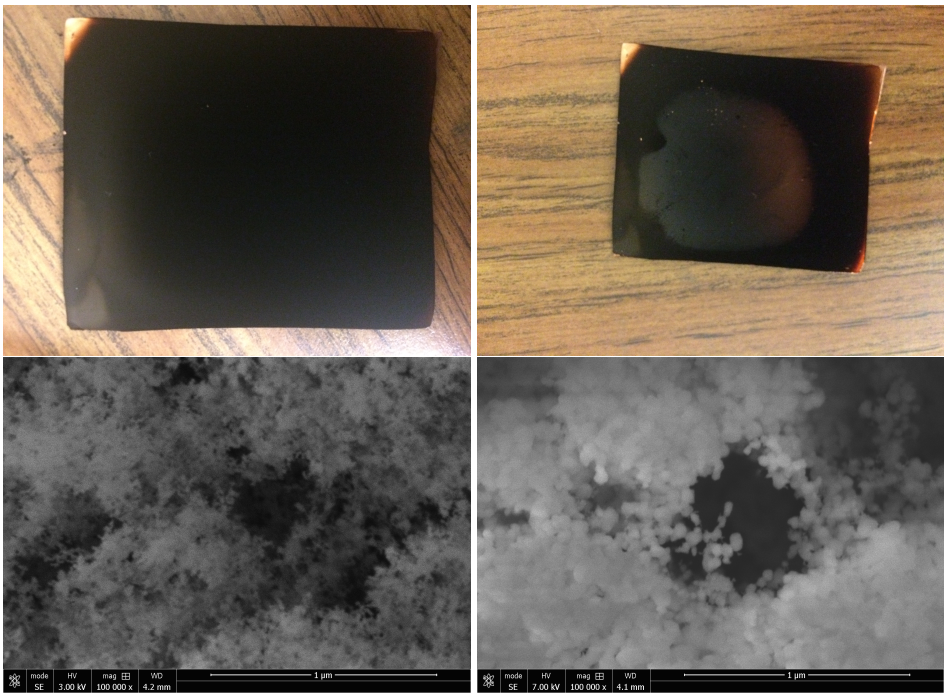

FIGURE 2.4: Images of a germanium film that was seemingly damaged by a camera flash. The top-left image was taken on August 18, 2016, while the top-right image is of the same film on August 23, 2016. The film was not used between these two exposures. Bottom: SEM images of an undamaged portion of this film (left) and a damaged portion of this film (right), both taken on September 12, 2016.

white; see Figure 2.3 for an image of such a damaged film deposited on copper. Subjecting a DLA film to optical intensities much greater than those used in our apparatus quickly damages the film; for instance, laser light with an intensity of roughly 10-100 $\mathrm{kW/m^2}$ will rapidly damage the films. There has also been one observed case in which a germanium film appeared to be damaged by the flash of a camera; see Figure 2.4 Although the effects of this damage on the photoacoustic response of the films has not been rigorously studied, anecdotally I have observed that the photoacoustic response is reduced but by no means eliminated.

Minimum Frequency

The process of generating photoacoustic sound signals from a DLA film requires that the film's temperature, and hence its density, be able to fluctuate between its two points of equilibria - the equilibrium state when the LEDs are off, and the equilibrium state when the LEDs are on. At high frequencies, this does not pose much of a constraint, as the LEDs switch between their on and off positions quickly enough that equilibrium cannot be reached. However, at low frequencies, it becomes possible for the DLA film and nearby air to nearly achieve equilibrium in the time that an LED remains on or off, at which point the temperature and density of the film will not change further until the LED switches states. This effect should significantly reduce the strength of signals at low frequencies. See Figure 2.5 for a plot of the photoacoustic response of a bismuth film with respect to the modulation frequency of the illuminating light; note the significant decrease in amplitude for frequencies below 9-10 kHz. Below this minimum frequency, one would expect that the sound field produced consists of a series of peaks, with positive peaks occurring when the

light is turned on and negative peaks corresponding to the light being turned off. The photoacoustic signal shown in Figure 2.2 demonstrates this nicely, as although the illumination is a plain square wave with frequency 250 Hz, the resulting pressure is a series of peaks (ignoring the subsequent ringing of the microphone), as we would suspect for sound below the minimum frequency of these films.

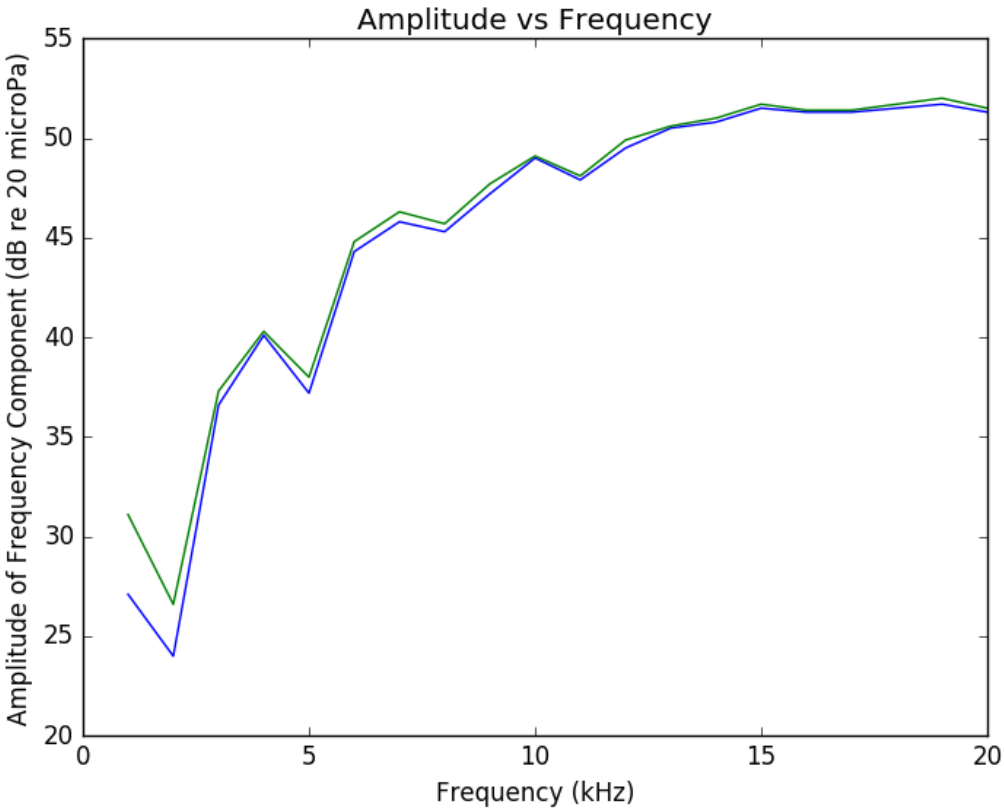

FIGURE 2.5: A rough plot of the photoacoustic repsonse of a bismuth DLA film as a function of frequency.

Chapter 3

Patterned Ultrasound Generation via Photoacoustic Effect

We have now established that diffusion-limited aggregate films are capable of producing rather strong sound signals at frequencies well above the threshold of human hearing. However, for many applications it is not enough to simply produce a strong sound field at a particular frequency; for instance, some applications may call for a particular intensity pattern, such that the ultrasound is "focused" in a particular region. In this chapter, we discuss the theory behind a phased array, a device which ideally would allow for the production of virtually-arbitrary sound fields. We also describe the optically-addressed photoacoustic phased array which we used to produce various ultrasound fields.

3.1 Phased Array Theory

A phased array is a collection of individual receiving or transmitting elements in which there is a phase shift between the signals received or transmitted by different elements. We start our explanation of the method of operation of phased arrays with the perhaps somewhat simpler case of a receiving phased array.

Suppose one has an arbitrary pressure field $p(\mathbf{r}, t)$ that was generated by an acoustic source, and one wanted to determine the source using a single microphone. Some information about the pressure field could be determined by using the microphone to make a measurement of the pressure field at a single point- for instance, Fourier analysis could determine the frequency components of the pressure field. However, this information would be insufficient to completely determine the source of the field; for instance, even if it is known that the acoustic source was a point source, one measurement would not be enough to distinguish between a weak pressure oscillation from a source near the microphone and a strong pressure oscillation from a source far from the microphone. By moving the microphone between measurements, one can compare the signals received and get some information about the source's location; for example, if the signal is stronger at one point than another, in most scenarios it is safe to assume that the stronger point is closer to the source, and the rate at which the pressure oscillation's amplitude changes can indicate the distance or shape of the source. However, this means of analysis only truly works under the assumption that the field that is being measured is static on time scales comparable to the length between measurements.

Now suppose that, instead of a single microphone, one had a number of microphones with which the pressure field can be measured. By placing each of these microphones in a different position and taking multiple measurements at the same time, one can glean much more information about the source than one could with a single microphone. There are many different methods that could be used to determine this information; for instance, some phased arrays function by averaging the signals from each of the microphones, which

results in a microphone that is very sensitive to sound propagating in a specific direction, for which the sound is in phase at all of the microphones, and is much less sensitive in others. In general, a phased array gives some information about how the sound field varies over the space occupied by the phased array; ideally, one could create a planar phased array that determines the pressure field and its normal derivative at all points within that plane with arbitrary precision, and determine the source from this information. [5]. Incidentally, this is one reason that having two ears rather than one is a benefit, since the signals received by each ear can be compared to determine the source of sound.

A transmitting phased array could be viewed as a time reversal of a receiving phased array. Instead of measuring the pressure field at every point in a plane, an ideal planar transmitting phased array can be used to create the desired pressure field at every point in a plane. Since knowing the pressure field and its gradient on all points on this plane uniquely determines the source of the sound, producing the time-reversed pressure field at all points on this plane creates a uniquely-determined region of high acoustic intensity, which we will colloquially refer to as a "focus" or "image". This suggests a method for creating arbitrary sound foci: the pressure field that the transmitting array must source can be determined by treating the desired image as an acoustic source, calculating the pressure field that would be measured by a receiving phased array at the same position as the transmitting phased array, and then time-reversing the result.

Another useful way of viewing the action of a transmitting phased array, which is the perspective typically used on this project, is that the phased array simulates the action of a lens. In optics, a lens functions by adding a position-dependent phase shift to light rays passing through it parallel to the optical axis; for instance, a plano-concave spherical lens shifts the phase of a light ray entering the lens from the planar surface at a point far from the optical axis by more than such a ray entering at a point near the optical axis, since the former has to pass through a larger distance within the lens, which has an index of refraction larger than that of air. If one were to look at the phase of the light produced as a function of position in a plane parallel to the planar surface of the lens, one would see that the phase increases as a function of the distance from the optical axis. With a phased array, a plane wave with this radially-increasing phase pattern can be produced directly by simply programming the phases into each of the individual transmitting elements. For a point further along the optical axis, the light field measured should be the same regardless of whether the light was created by passing a plane wave of light through a lens or generated by an ideal phased array. Using this perspective, if one knows of an optical lens that generates an intensity pattern, one can replicate that intensity pattern with sound by determining how the lens shifts the phase of light rays traveling parallel to the optical axis as a function of position and programming the phases of the transmitters to match this pattern.

3.2 Experimental Apparatus

Our transmitting phased array consists of a grid of light-emitting diodes (LEDs) controlled by a field-programmable gate array (FPGA) illuminating a DLA film. We will start our discussion by describing the operation of the FPGA. A field-programmable gate array is essentially a set of electronic switches, the positions of which can be changed based on programmable parameters. Each of the switches, or gates, outputs a voltage when it is "open" and does not put out a voltage when it is "closed". The voltage output by each of these gates controls a high-current switch, and current is allowed to flow through the LED when this switch is activated; effectively, when one of the FPGA's gates is open, current flows through the corresponding LED. By rapidly switching the corresponding gate between its

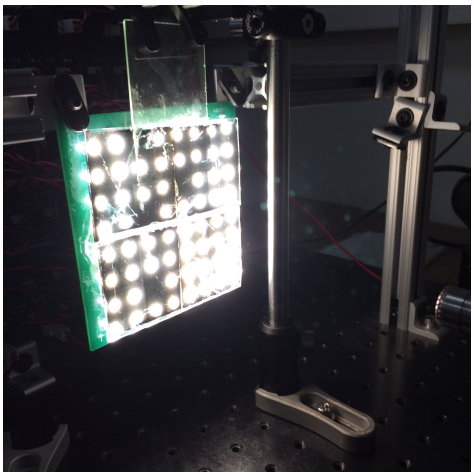

FIGURE 3.1: A photograph of our phased array in action.

open and closed positions, an LED can be quickly turned on and off, producing an approximate square wave of light. The FPGA has a clock rate of 50 MHz, and is thus capable of producing amplitude-modulated light at frequencies well above those typically used for this research, which did not exceed 200 kHz. The LEDs form an 8x8 grid, with half an inch between adjacent LEDs. A DLA film is suspended in front of the grid of LEDs. The film is typically deposited on a transparent surface, such as a glass slide or an optical component, with the coated side facing away from the LEDs. The light from the LEDs passes through the transparent surface and is absorbed by the film, and the photoacoustic signal produced by the film travels away from the LEDs.

This experimental apparatus functions as a phased array for sound and ultrasound. This phased array distinguishes itself from most other phased array technologies in that it does not require any piezoelectric transducers or magnets, instead simply requiring a light source. The optically-addressed nature of this phased array offers a number of benefits over typical phased arrays. For instance, the properties of the individual sources can be determined entirely by the light source, without the need for any mechanical changes to the system. When combined with the broadband nature of the photoacoustic response of DLA films, this phased array serves as a uniquely versatile source of sound and ultrasound. However, like virtually all phased array technologies, it deviates from an ideal phased array due to the following limitations:

- The phased array is not an infinite plane. This essentially limits the maximum distance of the "image" from the phased array; the array looks like a point source at long enough distances, which limits the variation in phase between a given LED and any point near the intended focus.

- The pressure field produced by the phased array is of limited resolution. This is a consequence of the fact that we have discrete optical elements (LEDs) rather than a continuous optical element over which we have arbitrary control. Additionally, the patches of the film illuminated by the LEDs do not fully cover the surface of the film, and are better approximated by point sources. Both of these limitations should effectively place a limit on the resolution of our "image".

- The derivative of the pressure field produced at a given point on a DLA film is normal to the film's surface. This is a consequence of the fact that the sound is generated by expansion of the film and nearby air, and symmetry dictates that this expansion should be along the normal axis of the film. So long as the intended image is in the

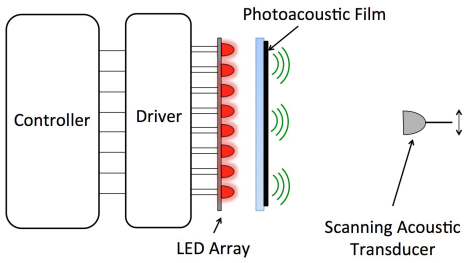

FIGURE 3.2: A schematic representation of our experimental setup.

far-field of the array and is not too far from the central axis of the array, this is not a particularly strong limitation when considering a planar DLA film, since the derivative of the pressure field generated by such a source would be approximately parallel to the central axis anyway.

- The relative amplitude of the pressure oscillations produced by each of the LEDs is not controlled. Up to any imperfections in the system, each LED transfers the same amount of optical power to the illuminated regions of the film, and the same amplitude signal is produced from each region (assuming each LED switches at the same frequency). Once again, this is not a strong limitation if the image is in the far-field and near the central axis, where the amplitude will not vary greatly over the phased array anyway.

To measure the sound fields we produce, we use a microprocessor-controlled motorized scanning acoustic transducer. This measurement system consists of a microphone mounted to a pair of translation stages. The translation stages are each connected to a 100 turns-per-inch turn screw driven by a stepper motor. The stepper motors are controlled by an Arduino, which allows one to control the position of the microphone through a computer connected to the Arduino; the stepper motors allow the microphone to scan over a region of about $(1.8\text{in})^2$. The Arduino is also connected to the LEDs through a driving circuit (which is discussed in more detail in Krutik Patel's thesis [4]), which allows the Arduino program to control when the LEDs are provided power. This setup allows one to automate the process of moving the microphone through a large number of positions when recording a large set of data. The driving circuit is also connected to both channels of a power supply, which provide power for both the LEDs and the stepper motors; both channels are set to supply power at about 8.0 V.

The data from the microphone is fed through a pre-amplifier (typically a Princeton Applied Research Model 113 Pre-Amp, although a Stanford Research Systems Model SR560 has also been used) and then sent to an oscilloscope. A ThorLabs PDA100A photodetector is also connected to the oscilloscope and is used for triggering. The oscilloscope is capable of recording and storing data on a flash drive; however, it should be noted that the oscilloscope records data at a rather slow rate, saving only one sample of data every three seconds. Replacing the oscilloscope with a data acquisition system capable of recording data more quickly would vastly improve data collection times. During recording, the oscilloscope is also connected to a pin on the Arduino that is programmed to output a set voltage for a period of time starting shortly after the LEDs turn on and stopping shortly before the LEDs turn off. The sum of this signal and the microphone's signal are recorded, and the average of the data sample can be used to determine if the sample should be treated as a data point or discarded.

3.2.1 Troubleshooting/Common Problems

There are a number of small problems that frequently occur when using the experimental apparatus described above. Some of these issues and common fixes are listed below:

- **If the LEDs won't turn on or output particularly dim light**:

  First, check that the two power supplies that are connected directly to the LED grid are both on. Then, check that the channel of the power supply controlling the LEDs through the driving circuit is set to output 8.0 V. If this is the case, check the driving circuit to see if any connections have become loose; in particular check the points where the circuit is connected to the power supply, the Arduino, or the LEDs. If this does not fix the problem, try moving the Arduino, making sure that the strain experienced by the Arduino and any connections made to it is minimal.

- **If the stepper motors fail to move or seem to "stick" at some point during a full rotation**:

  First check that the stepper motor in question is properly aligned with the turn screw. During operation, it is possible for the stepper motors to shift slightly, creating an angle between the turn screw and the motor; readjusting the motor to eliminate this angle can improve the motor's performance. If this fails, one should check the connection between the stepper motor and the turn screw. This connection is formed by a small metallic cube with two set screws in it connecting the motor and a broken-off hex key; this connection is particularly weak, and is typically the cause for the motor's failure to operate properly. Tightening the set screws or reforming the connection entirely should improve performance. If this also doesn't work, try spraying the turn screws with WD-40.

3.3 FPGA Programming

As described above, the sound field that is produced ultimately depends on the phase configuration that is set for the LEDs by the FPGA's programming; thus it is important to understand how to determine the phase pattern required and program the FPGA with this information. The general methodology has been outlined at the beginning of this chapter: one must either determine the phase shifts that model an optical lens that produces the desired intensity pattern (which is what was typically done on this project) or determine the phase pattern for the desired intensity pattern based on time-reversal. Here we will discuss the process of setting the required phases for the LEDs to generate a desired sound field. This section will primarily concern itself with a series of programs written to program the FPGA with the correct phases; the programs in question can be found in Appendix C.

The phased array is controlled via a Verilog program called phased_array_control.v. The program first creates an LED template signal based on a counter; the counter counts down at 50 MHz to 0 then resets to an initial maximum value, and the LEDs switch from on to off when the counter resets and when the counter crosses a threshold value. The maximum value of the counter determines the frequency of amplitude modulation; if the maximum value of the counter is n, then the frequency of amplitude modulation is $\frac{50\text{MHz}}{n+1}$ (the $+1$ accounts for the fact that the counter can have value 0). The threshold value of the counter is determined by the maximum counter value and the duty cycle of the desired light source; under typical circumstances, a duty cycle of 50% is desired, meaning that the LEDs are on and off for equal periods of time, and in this case the threshold value is just half the maximum value. Each of the LEDs is then assigned a number from 0 to 99 that represents that LEDs initial position in the template signal; in other words, these numbers determine

the phase of each LED in reference to this template signal. To change the pressure field generated by the phased array, one can change the maximum counter value and LED phase numbers manually, then recompile the program and upload it to the FPGA through the Quartus software; it should be noted that driver signature enforcement should be disabled on the computer used to upload the program so that the FPGA is recognized. Alternatively, one can store multiple different phase configurations on the FPGA at the same time and switch between them using the numerous switches on the FPGA.

In order to eliminate the need to calculate the proper phases of each of the LEDs and edit the Verilog program by hand, a number of Python programs were written to update this program automatically. These programs start with several lines of assignments which can be edited to determine various parameters of the intended pressure field, such as the position, frequency, and shape of the focus. If one edits these lines appropriately and then runs the program, this program will calculate the correct phase pattern for the LEDs and update the Verilog file so that the desired sound field is generated, such that one simply needs to compile and upload the updated Verilog code after running the Python script. Multiple programs that follow this basic routine have been created, including programs that only create specific sound fields (e.g. there is a program that will only generate line-shaped intensity patterns) and programs that allow one to specify the shape of the sound field within the program; although the latter is typically used, the programs for specific sound fields are often simpler due to the reduced complexity of the required input and are thus maintained. The types of sound fields that can be generated by the current versions of these programs are detailed in the Results section of this chapter.

3.4 Measuring Sound Fields

As previously mentioned, the measurement apparatus consists of a microphone mounted to a set of translation stages controlled by an Arduino and an oscilloscope. Here we outline in more detail the methodology behind using this apparatus to measure sound fields.

To measure the pressure field at a specific point, we consider the mean square of the signal which is measured by a microphone, increased by a preamplifier, then displayed and recorded by an oscilloscope. For measuring ultrasound fields, we used one of two microphones depending on the frequency. For 40 kHz ultrasound, we used a TCT40-16R ultrasonic transducer, and for 200 kHz ultrasound, we used a SensComp 200KHF18 ultrasonic transducer. Both of these microphones serve as narrowband detectors for ultrasound at their respective frequencies. The TCT40-16R has a nominal sensitivity of -65 dB re 1 microbar, although we estimate the sensitivity as -60 dB in our apparatus due to the large impedance of the oscilloscope and preamplifier. In either case, a number of measurements are taken and recorded, and a Python program determines the mean square of the signal averaged over all measurements.

In order to facilitate measurement of pressure fields at a large number of points, the microphone is mounted on a pair of translation stages. These translation stages allow for movement within a plane parallel to the plane of the LEDs. The translation stages are each controlled by a stepper motor and a turn screw. The turn screws allow for each translation stage to move over a range of about 1.8 inches, with 100 turns of the screw corresponding to 1 inch. The stepper motors, when controlled by an Arduino, allow for the translation stages to be moved automatically, which allows one to use an Arduino program to move to and record data at a large number of points.

It should be noted that the speed of data collection is severly limited by the slow rate at which the oscilloscope records data. The oscilloscope only makes a data recording about once every 3 seconds, and one typically wants several recordings per point so as to reduce

error. The delay between recordings is also non-constant, which introduces a problem with data collection for which the solution requires that data collection take even longer: there is no reliable way to tell just from a given data file's name or time stamp if the data contained in that file was recorded at the same position as was the previous data file or if it was collected at the subsequent point. Replacing the oscilloscope with a faster and more consistent data acquisition system would significantly improve the data collection process; however, in order to resolve the issue and make data collection with the oscilloscope possible, the following data collection process is followed:

Three signals in total are sent to the oscilloscope: the microphone signal (Channel 1), the digital signal from a pin on the Arduino (Channel 2), and the signal from a photodiode facing the LEDs (External Trigger). When the microphone reaches a position at which data is to be collected, the LEDs turn on. The signal from the photodiode triggers the oscilloscope, which causes it to start recording data. The recorded signal is the sum of the microphone signal and the signal from the Arduino pin; at this time, the latter should be 0. After a short period of time that is long enough to ensure that the oscilloscope has made at least one recording, the signal from the Arduino pin turns on. After a length of time sufficient to record the desired amount of data, both the LEDs and the signal from the Arduino turn off, and the microphone is moved again after a very brief pause. During data analysis, the average of each recording, which should be equivalent to the digital signal from the Arduino, is checked; if the average is above a threshold value, then the microphone must not have moved between the recording being considered and the previous recording, while if the average is below the threshold, one concludes that the microphone has moved, notes that the next recording with an average above the threshold must be the first recording at the next position, then discards the recording.

3.5 Results

We start with the simple case of a DLA film deposited on a spherical surface. By definition, every point on this surface is equidistant from the center of curvature of the surface; therefore, the sound from a point source located at the center of curvature would be in phase at all points on the surface, with the pressure gradient normal to the surface and of equal magnitude at all points on the surface. Thus, if we want to use a DLA film deposited on a spherical surface to create sound that is focused at the center of curvature of the surface, we should ensure that the sound produced at each point on the surface is in phase. This is achieved to good approximation (i.e. ignoring the very slight phase shift due to the light having to travel farther to points further from the center of the curved surface) by simply illuminating the surface with a uniform plane wave. The result, as expected, is a beam of sound focused at the desired point. Figure 3.3 shows line scans of the focus produced in this manner at 40 kHz and 200 kHz and compare the data to a diffraction-limited spot of the same frequency and amplitude. Note that the 40 kHz focus is about a factor of 5 wider than the 200 kHz focus. Typical RMS pressures for the sound fields measured at 40 kHz are on the order of 0.1 Pa, although most data was collected and plotted in arbitrary units.

Now suppose one wanted to generate sound that is focused at an arbitrary point with coordinates $(x_s,\ y_s,\ z_s)$, using a DLA film deposited on a planar surface in the x-y plane with its center at the origin. We start by imagining that we have a point source of sound at the desired location, and then calculate the phase of the sound produced at each point in the x-y plane; we do not bother calculating the amplitude of the pressure oscillation or the direction of its gradient, although we note that both are approximately uniform so long as we resrict our attention to the region where $z \gg x,\ y$. The distance from the source to an arbitrary point on the surface is given by $d = \sqrt{(x - x_s)^2 + (y - y_s)^2 + z^2}$. Dividing this

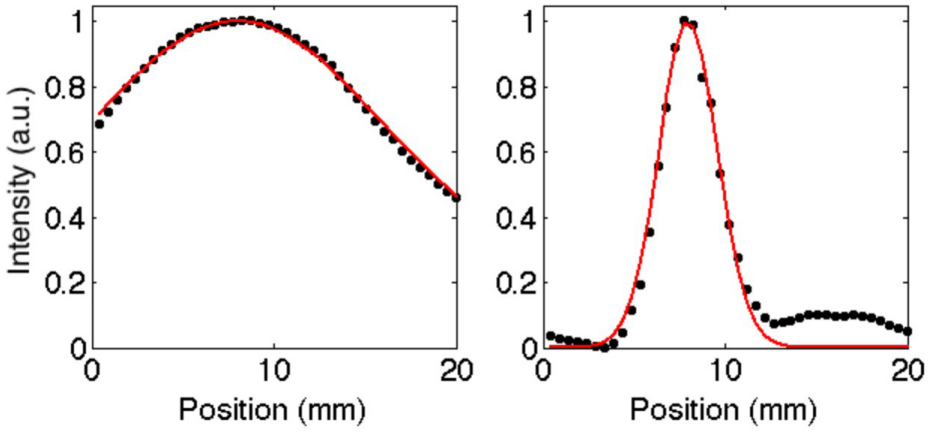

FIGURE 3.3: Cross-sections of the sound pattern produced by a DLA film deposited on a lens when illuminated by a plane wave. Results on the left are for 40kHz, results on the right for 200kHz. The data points show the actual data collected, while the solid lines plot the shape of a diffraction-limited focus with the same amplitude.

distance by the wavelength $\lambda = \frac{c_s}{\nu}$ gives the phase difference between the source and the point in the plane. Time-reversing this result, which consists of negating this phase, gives the phase of the sound that should be produced at a given point in the plane to ensure that the sound is in phase at the desired point. We consider the results modulo 2π (since a phase shift of 2π is irrelevant) and program the phased array to produce sound with the resulting phase at each point. Note that the programmed phases are approximate; the phase is represented as a number from 0 to 99 that increments every $\frac{1}{100}$ of a period, although this phase resolution is determined primarily by the program used to run the FPGA, and could easily be increased so long as clock rate of 50 MHz is not exceeded. Figure 3.4 shows data for a number of foci produced at different points; note that the origin is at the same position in all data runs.

The process outlined above allows one to use the phased array to generate a focus at an arbitrary point; however, more complex patterns may also be created. For instance, a linear focus can be created by having the phase shift depend on the distance from a given line as opposed to a single point (for instance, dropping the dependence on x to create a line corresponding to the x-axis in the plane). Such a linear focus is shown in Figure 3.5. It should be noted that the best results are produced when the line is question is diagonal; this may due to limitations of our phased array, since using this method, all of the LEDs in a given column are shifted so as to produce a focus at the same point, leading to a series of 8 distinct spots, whereas there are more distinct points at which multiple LEDs are in phase if the line is vertical. One can also choose a phase pattern that creates a half-ring focus by basing the phase of the LEDs on the distance to the nearest point on the desired half-ring.

For more complex patterns, multiplexing techniques can be used to generate images corresponding to the combination of two different foci. There are two types of multiplexing: spatial multiplexing and time multiplexing.

With spatial multiplexing, the LEDs are split into a number of subgroups, and each subgroup is programmed to generate a different foci. For instance, as shown in Figure 3.6 the LEDs can be split into two groups in a checkerboard pattern and each group can be programmed to generate a focus at a different point, resulting in a sound field focused at two points. In general, when using space multiplexing one must consider the effects of interference between the sound from the different groups on the final sound field; however, experimentally we find that interference does not significantly alter the final image for the

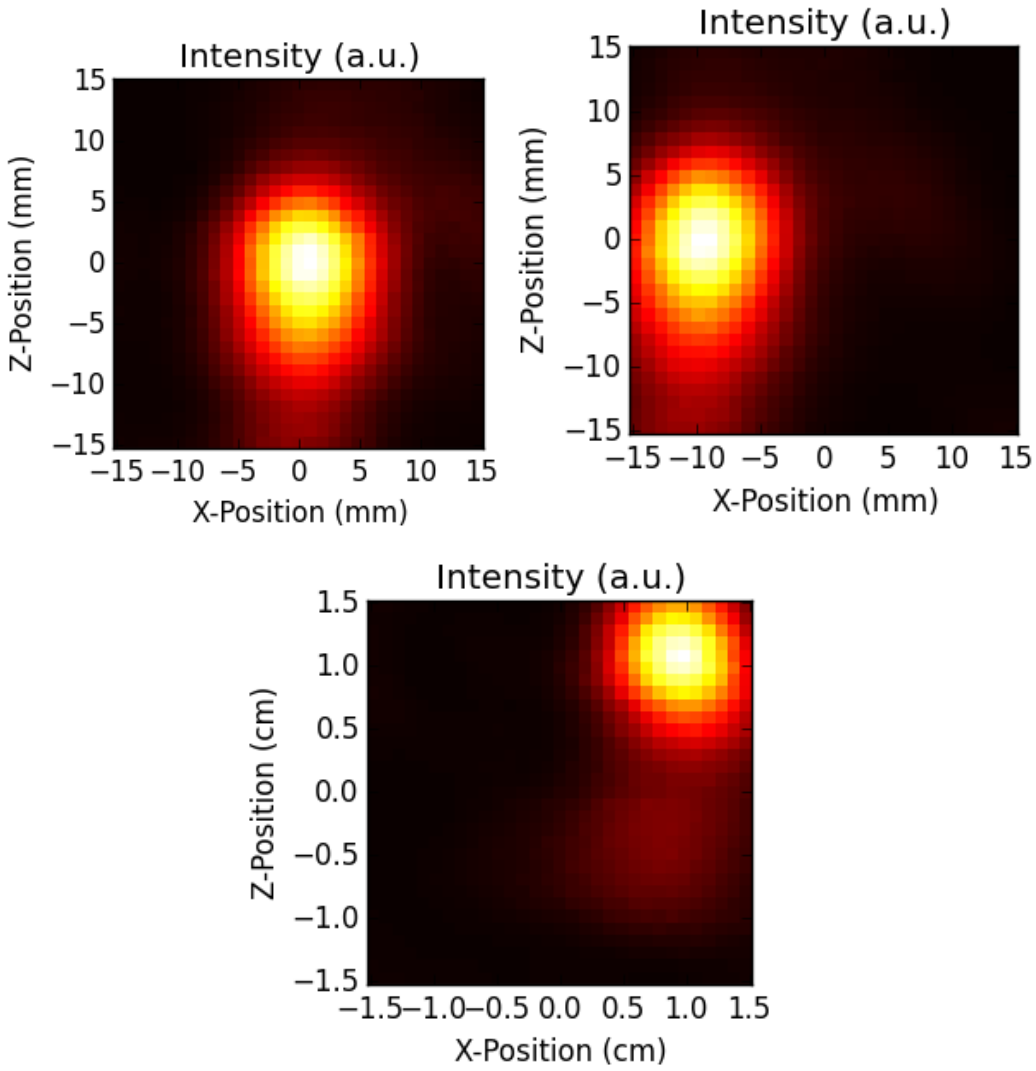

FIGURE 3.4: Three sets of data showing a focus that can be moved to various points. Data shown is sound intensity as a function of position as measured by a scanning acoustic transducer.

case of two foci.

When using time multiplexing, the LEDs are programed to quickly switch between multiple phase patterns. So long as the time that it takes to cycle through all programmed phase patterns is much smaller than the time scales relevant for the application in question, the sound intensity pattern is the same as the average of all those generated by the separate phase patterns. Unlike space multiplexing, time multiplexing generally escapes any interference effects, allowing one to more reliably specify a complex intensity pattern by combining more simple intensity patterns. Some examples of sound fields generated by time-multiplexing are shown in Figures 3.7 and 3.8 Note that, although this has not been tested, time-multiplexing may fail if the amount of time spent producing any individual sound field is less than one period at the desired frequency, since there would be no way to determine the true frequency at any given point on the film since only square waves of light are used.

3.6 Photoacoustic Efficiency

One relevant parameter for characterizing the performance of a transducer is the efficiency with which it converts light to sound. We estimate that the efficiency with which a bismuth film produces 40 kHz ultrasound is on the order of 10^{-8}. As this estimate is rather involved, the process used to reach this estimate is detailed below.

Calculating the efficiency of the DLA films consists of three main steps: calculating the amount of light energy that is radiated by an LED, measuring the sound field generated when a DLA film is illuminated by this LED, and calculating the energy in the measured sound field. If we take the square root of the numbers used to generate images of the sound field, we get the rms voltage displayed on the oscilloscope. We use a preamplifier to amplify the signal measured by the microphone, and dividing by the preamplifier's gain (2000 for 40 kHz sound) yields the rms voltage output by the microphone. As mentioned above, we estimate the microphone's sensitivity as -60 dB re 1 $\frac{\mathrm{V}}{\mu\mathrm{bar}}$, or $10^{-2}\frac{\mathrm{V}}{\mathrm{Pa}}$; dividing the voltage by this number gives the rms pressure of the sound field. The power flowing through a region

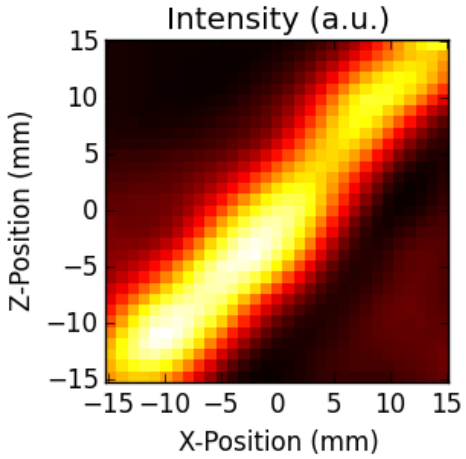

FIGURE 3.5: A linear sound field.

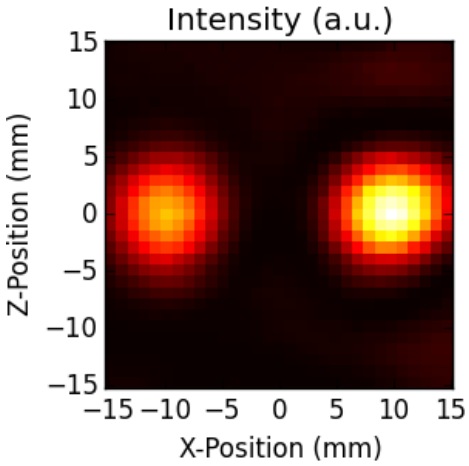

FIGURE 3.6: A pair of foci, demonstrating spatial multiplexing.

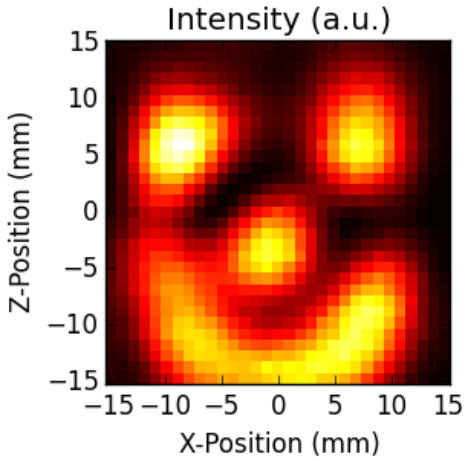

FIGURE 3.7: Data from a sound pattern that demonstrates the time-multiplexing technique. By alternating between producing a pair of foci and a semicircular pattern, a sound pattern the combination of the two on timescales longer than the time it takes to switch between the two patterns is generated.

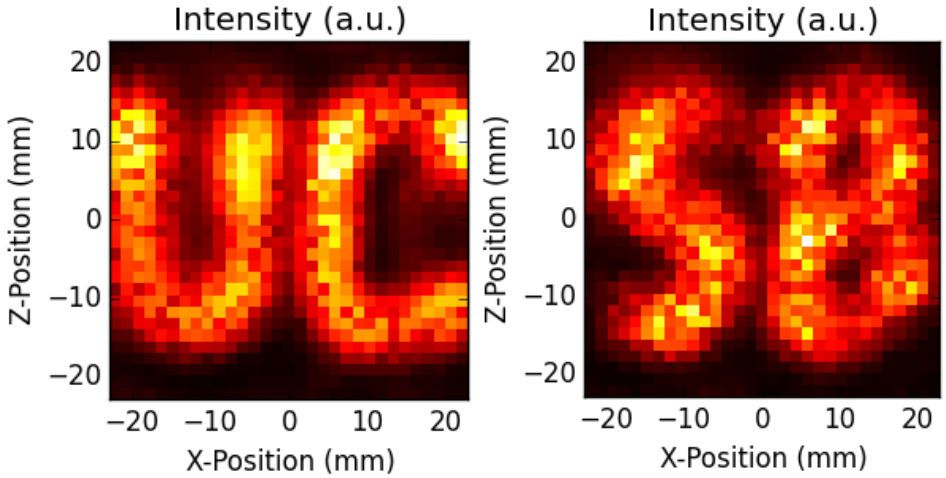

FIGURE 3.8: Data from two different sound patterns demonstrating the time multiplexing technique. Both scans were taken at 40kHz

of area A in a sound wave is then given by

$$P = \frac{p_{rms}^2 A}{\rho v} \tag{3.1}$$

where ρ is the density of air and v is the speed of sound in air. We assume that

a) The portion of the sound wave that does not travel through the region measured is of negligible power, and

b) Any directionality of the microphone's sensitivity is negligible at the points measured.

Under these assumptions, the total power of the sound wave is simply the sum of the powers corresponding to each data point.

To determine the electrical power used by the LED, we measure the voltage V across the

resistor that the LED discharges through and note its resistance R. Using these quantities, the electrical power is given by $\frac{V^2}{R}$ by Ohm's law. We then multiply this result by the electrical efficiency of the LED measured in lumens per watt; this quantity is given by the luminous output of the LED divided by both the maximum voltage drop across the LED and the test current [6], all of which are listed on the datasheet [7].

One then has the power of the emitted light in lumens, which is a unit of luminous power that is weighted by the eye's response to light; in order to obtain the actual power of the light, this weighting must be inverted. This is done by dividing the result by the luminous efficacy [8]; if $J(\lambda)$ is the spectral power distribution of the light source and $y(\lambda)$ is the luminosity function, which describes the eye's sensitivity to light at a given wavelength, then the luminous efficacy is given by

$$\frac{\int y(\lambda)J(\lambda)d\lambda}{\int J(\lambda)d\lambda} \tag{3.2}$$

The spectral power distribution of the LED is also on the data sheet, and we approximate it by the sum of two Gaussians with the same peak values and full widths at half maximum as shown in the plot. The luminosity function is also approximately a Gaussian, and is likewise approximated. The two integrals are calculated numerically using the approximate forms of $y(\lambda)$ and $J(\lambda)$ and used to determine the luminous efficacy, and thus the power of the emitted light. We then used an optical power meter to measure how much light was passing through the thin film without being absorbed; since the readout on the power meter when the film was present was a factor of about 0.74 of its value when the film was absent, we conclude that the film absorbs 26% of the light incident upon it (I should note that the film used for this was rather thin, which reduced the optical absorption). We multiply the power emitted by the LEDs by 0.26, then divide by a factor of 2 to account for the fact that the LEDs are off half of the time, to get the optical power absorbed by the DLA film. Dividing the result for the sound power by this number gives our efficiency of 10^{-8}.

While this may seem like a rather low efficiency, it should be noted that this is still quite a bit more efficient than many other photoacoustic sources. For instance, this efficiency is 3.5 orders of magnitude greater than that reported for a silicon wafer coated with metal in air [9], and is comparable to that of metallic films [10] and nanotube-based structures [11] in water despite the greater efficiencies possible in water compared to air.

Chapter 4

Nonlinear Acoustics and the Diffraction Limit

Although the phased array discussed in the previous chapter allows us to generate a wide variety of sound fields, there are some limits to the types of sound fields that can be produced. In particular, the diffraction limit of sound in air imposes a limit on the spatial resolution of sound fields based on their frequency. Because this limit depends on the frequency of sound, one might hope that by taking advantage of nonlinear frequency mixing, one might be able to overcome or at least reduce this limit. Moreover, our optically-addressed phased array is capable of thoroughly exploring this problem due to the broadband nature of the photoacoustic response of DLA films. This chapter presents and discusses our efforts to utilize our phased array to generate sub-diffraction sound fields via nonlinear frequency mixing.

4.1 Background

4.1.1 Diffraction Limit

Based on the discussion thus far, one might be tempted to conclude that the experimental setup described in the previous chapter should allow for practically arbitrary acoustic or ultrasonic generation; while we are limited by issues such as the size and pitch of our phased array or the clock rate of the FPGA, one might think that a version of our setup without these limitations would be able to produce any sound field given the right phase configuration. This conclusion is false; the diffraction limit of sound in air poses a lower bound on the size of a focus that can be generated, and the direct production of ultrasound as discussed previously cannot produce a focus smaller than determined by this lower bound. The diffraction limit of sound is generally a function of a number of parameters, such as the frequency of the sound being produced, the dimensions of the sound source, and the distance from the sound source where the sound is focused. If one wished to make a general, order-of-magnitude approximation of the diffraction-limit, one would use the wavelength of sound,

$$\lambda = \frac{v_s}{f} \tag{4.1}$$

However, to get a more accurate estimate of the diffraction-limit, we must consider the details of our phased array.

Suppose the phased array is programmed to produce a focus at frequency f at a position $\mathbf{r} = (x, y, z)$, where the origin is the center of our phased array. We can describe our system to a good approximation by treating the sound produced by the phased array as a Gaussian beam, the beam waist of which is located at $\mathbf{r}$. We can then determine the thickness of the beam at the beam waist w_0 from analogous results in optics [12]: if F is the distance between

the phased array and the beam waist $|\mathbf{r}| \approx z$ and D is the diameter of the phased array, then

$$w_0 = \frac{2\lambda}{\pi}\frac{F}{D} \tag{4.2}$$

Given this result for w_0, the amplitude of the pressure oscillation at a position within the plane of the beam waist at a distance ρ from the optical axis is given by

$$P_s = Ae^{-\frac{\rho^2}{w_0^2}} \tag{4.3}$$

where A is the amplitude on the optical axis at the beam waist. The intensity of the sound is proportional to the square of the pressure oscillation; therefore, the ratio of the intensity at ρ compared to the intensity at the center of the beam waist is given by

$$\frac{I_\rho}{I_0} = e^{-\frac{2\rho^2}{w_0^2}} \tag{4.4}$$

One can use this result to find the full-width at half-maximum of the intensity pattern:

$$e^{-\frac{2\rho_{FWHM}^2}{w_0^2}} = \frac{1}{2} \tag{4.5}$$

$$\frac{2\rho_{FWHM}^2}{w_0^2} = \ln(2) \tag{4.6}$$

$$\rho_{FWHM} = w_0\sqrt{\frac{\ln(2)}{2}} = \sqrt{2\ln(2)}\frac{\lambda}{\pi}\frac{F}{D} \tag{4.7}$$

This formula allows us to get an approximate value for the diffraction limit for sound produced by our phased array. Setting $D = 4$ inches, we find that ρ is approximately related to the focal distance F (which is just the value of z at which the sound is focused) by

$$\rho = (3.69\ \mathrm{m}^{-1})\lambda F \tag{4.8}$$

Although this result is clearly an approximation - the sound produced by our system is not exactly a Gaussian beam, and there is some ambiguity in what value of D should be used for a non-circular source - this places a clear limit on the resolution of intensity patterns that we can produce.

4.1.2 Nonlinear Acoustic Generation

While the diffraction limit places a restriction on the resolution of intensity patterns that we can produce at any frequency, the restriction is much more severe for sound fields at low frequencies. Note that the wavelength of sound scales inversely with frequency, so the full-width at half max doubles when the frequency is cut in half. At a focal distance of 15cm, the full-width at half maximum of a 40kHz focus is 4.7 mm; for audio frequencies, which range from 20Hz to 20kHz, the width at the same focal distance ranges from 9.5 mm to 9.5 m. (although it should be noted that the approximation for ρ breaks down well before the beam waist thickness exceeds the source diameter). Large diffraction limits essentially eliminate any possible control over the sound field, and we are thus unable to produce arbitrary sound fields directly at low frequencies.

However, one might suspect that there is some method by which one can generate a sub-diffraction intensity pattern, which allows for a focus smaller than the limit derived above.

There are some known methods that allow for one to beat the diffraction limit, some of the most promising being those that depend on nonlinear interactions between sound fields at different frequencies [13]. The main effect of nonlinearity in the propagation of sound is to create frequency mixing - if one generates two sound waves, one with frequency f_1 and another with frequency f_2, nonlinear effects will create the sum and difference frequencies, $f_1 + f_2$ and $|f_1 - f_2|$.

More specifically, the behavior of sound in air up to second order in pressure is given by the Westervelt equation [14]:

$$\nabla^2 p - \frac{1}{v_s^2}\frac{\partial p}{\partial t} + \frac{\delta}{v_s^4}\frac{\partial^3 p}{\partial t^3} = -\frac{\beta}{\rho_0 v_s^4}\frac{\partial^2 p^2}{\partial t^2} \tag{4.9}$$

where δ is the sound diffusivity, β is the nonlinear coefficient of air, and ρ_0 is the density of air. Rather than solving this equation directly, we note that the equation depends on p^2. If we generate sound at frequencies f_1 and f_2, then p must have terms corresponding to waves at these frequencies. If we assume that these waves are harmonic in nature, then at a given point,

$$p = A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t) + \text{ some other terms} \tag{4.10}$$

$$p^2 = A_1^2 \cos^2(\omega_1 t) + A_2^2 \cos^2(\omega_2 t) + 2A_1 A_2 \cos(\omega_1 t)\cos(\omega_2 t) + \text{ some other terms} \tag{4.11}$$

Using a trigonometic identity, we note that the cross term can be rewritten:

$$2A_1 A_2 \cos(\omega_1 t)\cos(\omega_2 t) = A_1 A_2(\cos((\omega_1 + \omega_2)t) + \cos((\omega_1 - \omega_2)t)) \tag{4.12}$$

The second time derivative term thus introduces terms with the frequencies $f_1 + f_2$ and $|f_1 - f_2|$. In order for the equation to hold true, the other terms in the equation must also have components at these frequencies so as to cancel out these terms; this requires that p have components at these frequencies. Therefore, if we try to produce sound at two frequencies in the same region of space, we also produce sound at the sum and difference frequencies.

Note that the amplitudes of the generated nonlinear signals, or at least that of the terms generated by the p^2 term, seem to depend on the product of the amplitudes of the primary signals. This could, perhaps, be taken advantage of to produce subdiffraction sound fields. For the sake of argument, we will assume that the intensity of the difference frequency sound produced at a given point is directly proportional to the product of the primary signal amplitudes at that point. Continuing our earlier approximation that the sound generated by the phased array is a Gaussian beam, we note that the amplitude of the nonlinear signal in the plane of the beam waist is of Gaussian form:

$$I \propto e^{-\frac{\rho^2}{(w_0)_1^2} - \frac{\rho^2}{(w_0)_2^2}} \tag{4.13}$$

If we assume that $\frac{1}{(w_0)_1^2} \approx \frac{1}{(w_0)_2^2}$, then the full-width at half-maximum of the intensity pattern of the nonlinear signals is just the full-width at half-maximum of the primary signals. This means, for example, that one could produce a focus of 20 Hz sound, which has a diffraction limit of 9.5 meters under our approximations and a focal length of 15 cm, by generating two ultrasonic signals around 40 kHz, where the diffraction limit at this focal length is 4.7 mm, and the 20 Hz focus would have a spot size of 4.7 mm, offering a drastic improvement over the diffraction limit. Additionally, it is possible to choose the primary frequencies such that both primary frequencies and the sum frequency are above the relevant bandwidth for a given application; for example, if one chooses two ultrasonic frequencies for the primary signals, then only the difference frequency is audible, so an audio microphone in the focal plane would only detect a focus at the difference frequency and below the diffraction

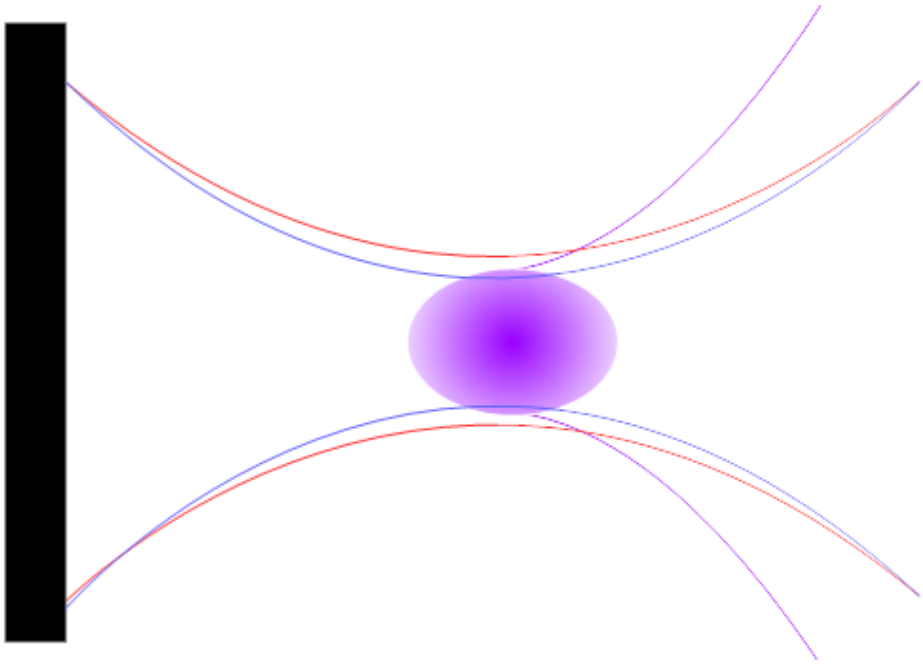

FIGURE 4.1: A diagram showing how one might generate sub-diffraction-limited foci by mixing two higher frequency foci. The two primary signals (red and blue) are Gaussian beams with beam waists at the same position. In regions near the beam waist, the two signals are strong enough that the two can mix, producing the difference frequency signal (purple). The difference signal then spreads as a Gaussian beam , with the mixing region (also purple) serving as the beam waist. Because the beam waist is more narrow than would normally be possible with the same transducer, the focus in the plane of the beam waist is diffraction-limited.

limit. This would allow us to extend our fine control of ultrasound as described in the previous chapter to sub-diffraction control of acoustic frequency sound fields, since we could produce nearly-arbitrary acoustic fields by overlapping various nearly-arbitrary ultrasound fields.

4.2 Experimental Setup

The experimental apparatus used to generate nonlinear signals is essentially the same as that used for direct ultrasound generation. The main difference is in the FPGA's programming; whereas in direct generation all of the LEDs are set to the same frequency, for this experiment we set half of the LEDs to one frequency and the other half to another. This is done in a similar manner as the spatial multiplexing used for ultrasound fields; instead of producing two foci at the same frequency and different locations, we produce foci at the same location and different frequencies. The DLA films are still either suspended in front of the phased array or deposited on a lens which is then placed in front of the phased array. We also ordered a large mirror blank to use as a large lens for this purpose; the lens is 6" in diameter and has a focal length (as a mirror) of 6", corresponding to a center of curvature 1 foot away from the lens.

Since the microphones used for the previous experiment, which were only capable of measuring specific ultrasonic frequencies, could not be used to detect frequencies in the audible range for this experiment, we had to obtain a microphone for the acoustic range to measure our sound fields. We initially tried using a number of cheap microphones intended

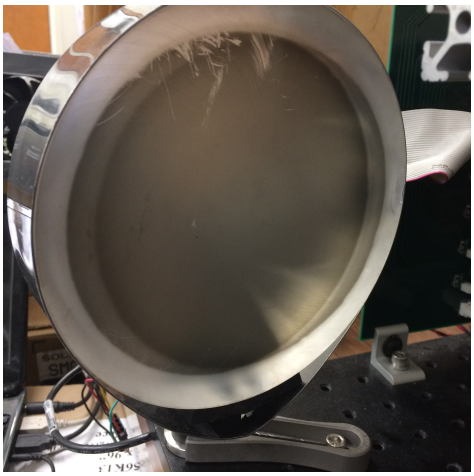

FIGURE 4.2: A large uncoated mirror blank that we've repurposed as a photoacoustic transducer by depositing a DLA film onto its surface. The radius of curvature of the lens is 1 foot.

for use with a computer, as these tended to be quite small, which should increase the resolution with which we could measure the sound field. However, a variety of issues with these microphones, such as low sensitivity, restricted bandwidth, high noise floor, difficulties installing software to analyze audio input on the lab computer, and questions of how to properly power the microphones if we wished to use lab equipment to measure the signal rather than a computer, made these efforts ineffective. We were eventually able to borrow a Bruel & Kjaer sound level meter with a 4189 microphone installed. This device allowed us to measure the individual frequency components of our sound fields in decibels.

4.3 Results

Although we could not initially detect a difference frequency signal generated by a planar surface, early experimentation using a DLA film deposited on a small lens allowed us to determine by ear that the difference frequency signal was being generated. Qualitatively, this sound field did appear to be rather tightly focused at the center of curvature of the lens, although it was also rather quiet; the low amplitude of the signal encouraged us to buy the large mirror blank, as this would allow us to use all of the light from the phased array to illuminate a curved surface.

Once we obtained the sound level meter, we were able to measure the spot size of the foci generated at the difference frequency. Such foci were found to be generated both when using the large mirror blank and when using a planar surface. Moreover, the location of the focus did correspond with the location that the FPGA was programmed to produce foci of the two primary frequencies; we were thus able to use the nonlinear mixing of sound in air to produce a focus at the difference frequency. However, all measured sound fields had a spot size comparable to the diffraction limit of sound. See, for instance, Figures 4.3, 4.4, and 4.5. The sound fields produced all have full widths at half maximum comparable to the diffraction limit when a width significantly smaller than the diffraction limit would be expected.

Suspecting that part of the problem might be that the subdiffraction difference frequency sound field being generated in the air was masked by difference frequency sound being generated in the film due to the regions lit by the LEDs overlapping, we also tried a number of

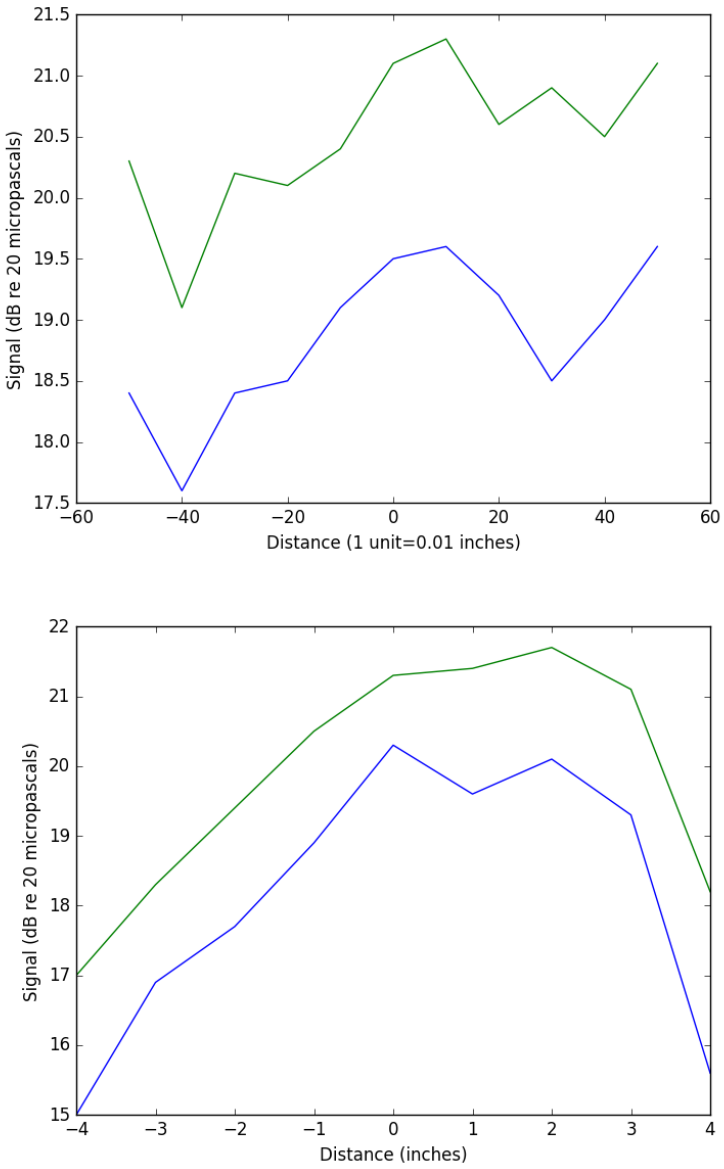

FIGURE 4.3: Line scans of a sound field generated by a DLA film deposited on the large mirror blank. The focal length was one foot, and the difference frequency was 4.963 kHz; the primary frequencies used were not recorded but can safely be assumed to be at least 40kHz. A number of measurements were taken at each point; the blue line represents the average amplitude, and the green line represents the maximum amplitude. The top plot shows a line scan over a very small range, while the bottom plot shows a line scan over a larger range. The full width at half maximum is over 6 inches, which is on the order of magnitude of the diffraction limit of 3.07 in.

different patterns for the LEDs. Typically, the LEDs were grouped in a checkerboard pattern, such that if a given LED was outputting light modulated at one of the two frequencies, all adjacent LEDs were outputting light modulated at the other frequency. This maximizes the potential for overlap between the regions illuminated by the LEDs; to reduce this overlapping, we sometimes switched to a pattern with the LEDs split into quadrants, with LEDs in the top-left or lower-right quadrants producing one frequency and the other quadrants producing the other. However, this did not seem to appreciably change the sound field,

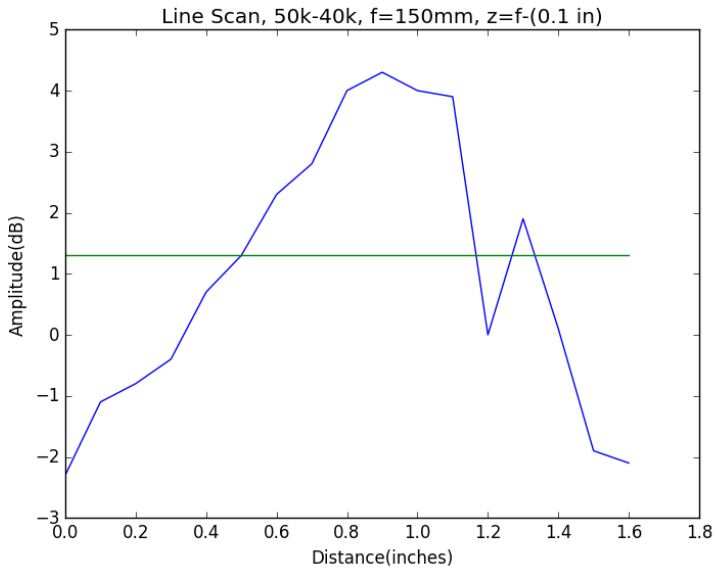

FIGURE 4.4: A line scan of a sound field generated by a planar source measured at 10 kHz when the FPGA was programmed to output sound at 40 kHz and 50 kHz. The distance from the film to the sound level meter was roughly 14.75 cm, and the intended location of the focus was 15cm from the FPGA. Note that the data point with value 0 does not represent a point where 0 was measured, but rather represents a data point that was accidentally omitted when measuring. The horizontal line indicates an amplitude 3dB below the maximum measured. The measured full width at half-max is roughly 0.8 inches; this is not below the diffraction limit for this frequency and focal length, which gives a spot size of 0.74 in.

which was measured to have a width comparable to that obtained from a checkerboard pattern. A variant on this pattern in which LEDs on the border between two quadrants were left off was also tried, but this only seemed to yield a decrease in amplitude. This data suggests that the sound field produced at the difference frequency may not be diffraction-limited.

4.3.1 Spot Size Scaling Test

Consider the intensity pattern in the focal plane where the two primary frequencies mix. Recall that our assumptions about the intensity pattern of the difference frequency sound led to the conclusion that, in the focal plane,

$$I \propto e^{-\frac{\rho^2}{(w_0)_1^2}-\frac{\rho^2}{(w_0)_2^2}} \tag{4.14}$$

We concluded earlier that, if the beam waists of the two primary signals are comparable, then the width of the difference frequency focus is the same as the width of the two primary foci; let's consider this more precisely, and find the full width at half-maximum of the difference focus:

$$\frac{1}{2} = e^{-\rho_{FWHM}^2(\frac{1}{(w_0)_1^2}+\frac{1}{(w_0)_2^2})} \tag{4.15}$$

$$\rho_{FWHM} = \sqrt{\ln(2)\frac{(w_0)_1^2(w_0)_2^2}{(w_0)_1^2+(w_0)_2^2}} \tag{4.16}$$

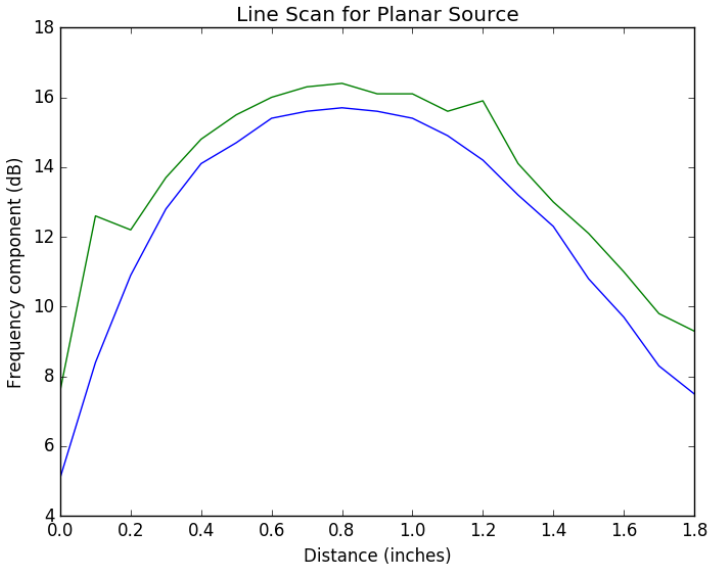

FIGURE 4.5: A line scan of a sound field measured at 10 kHz when the FPGA was programmed to output sound at 55 kHz and 65 kHz. The distance from the film to the sound level meter was roughly 7 cm. A number of measurements were taken at each point; the blue line represents the average amplitude, and the green line represents the maximum amplitude. The measured full width at half-max is roughly one inch, which is on the order of magnitude of the diffraction limit of 0.74 in.

We consider the derivative of ρ_{FWHM} with respect to $(w_0)_1$:

$$\frac{\partial \rho_{FWHM}}{\partial (w_0)_1} = \frac{1}{2}\sqrt{\frac{(w_0)_1^2 + (w_0)_2^2}{(w_0)_1^2 (w_0)_2^2}} \left(2\frac{(w_0)_1 (w_0)_2^2}{(w_0)_1^2 + (w_0)_2^2} - (w_0)_1^2 (w_0)_2^2 ((w_0)_1^2 + (w_0)_2^2)^{-2} (2(w_0)_1) \right) \tag{4.17}$$

For the sake of this argument, we only care about the sign of the derivative, so we are free to drop any strictly positive prefactors:

$$\frac{\partial \rho_{FWHM}}{\partial (w_0)_1} \propto (w_0)_1 (w_0)_2^2 ((w_0)_1^2 + (w_0)_2^2) - (w_0)_1^3 (w_0)_2^2 \tag{4.18}$$

$$\frac{\partial \rho_{FWHM}}{\partial (w_0)_1} \propto (w_0)_1 (w_0)_2^4 > 0 \tag{4.19}$$

Because this derivative is strictly positive, the width of the difference frequency spot must vary with the width of the primary frequency spots, so the difference frequency spot size should scale with the primary frequency wavelength. However, under normal (i.e. diffraction-limited) conditions, the width of the difference frequency spot should vary with the difference frequency wavelength.

The argument above suggests an experiment to test whether or not the sound field that we measure is diffraction-limited. Suppose we create a nonlinear signal using primary frequencies f_1 and f_2 with $f_2 > f_1$, and we measure the difference intensity pattern's width w. Suppose we then decrease f_1. This corresponds to an increase of $(w_0)_1$, which means that if our model is correct, the width of the difference sound field must increase. However, this also increases the frequency of the difference signal, decreasing its wavelength, so if the difference sound field width is found to decrease or not change at all, we can reasonably conclude that either our model is incorrect or the sound field is not sub-diffraction-limited.

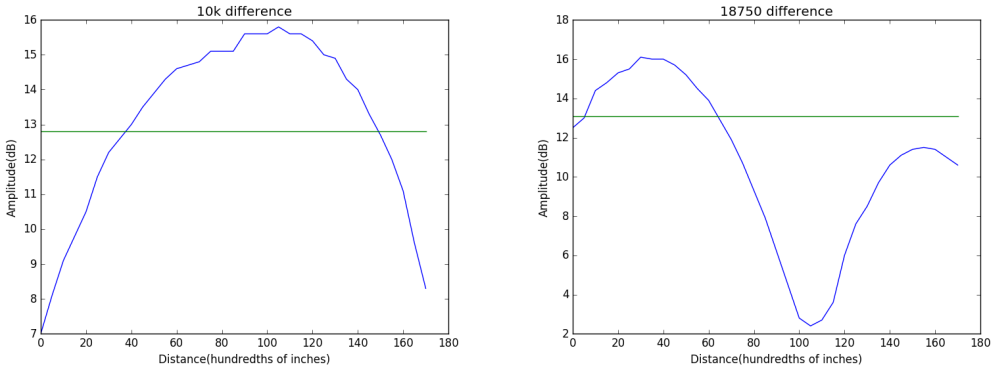

FIGURE 4.6: Data from the experiment described in the results section. For the first set of data, the primary signals had frequencies 40kHz and 50kHz; for the second set of data, the primary signals had frequencies 31.25 kHz and 50 kHz. The green line in each indicates an amplitude 3dB below the maximum measured amplitude; the points where the data crosses this line are used as a rough indicator of focus width.

This test was performed with our system; see Figure 4.6 for the results of this experiment. We can make at least two observations regarding this data:

1) While the first set of data is centered on the peak of the difference signal, the second set of data is centered on a local minimum. This is despite the fact that the two foci should be generated at the same position for both sets of data and the microphone was not moved between the two data sets (although I will note, for the sake of caution, that there is a possibility that the microphone was slightly and unintentionally rotated between the two data runs; I recall retaking this data and seeing the same thing, but did not find this other data, only finding a set of data that showed a monotonic decline in the 18.75k signal along the direction of the scan). This could be interpreted as a slight shift in the microphone's alignment or as a possible interference effect. If this is interpreted as an interference effect, then it could suggest that either our model of how the difference frequency is generated and spreads is incorrect, or that there are additional sources of sound at the difference frequency that are comparable to or stronger than the difference frequency generation in the focal plane.

2) Assuming that we do not interpret the minimum at the center of the 18.75 kHz line scan as an interference effect, we observe that the 3dB width of the peak in the second set of data appears to be quite a bit smaller than that of the first set of data, from which we would conclude that our explanation for the generation of nonlinear signals is incorrect.

4.4 Alternative Models and Suspected Problems

There are a number of possible explanations for why we have thus far not detected sub-diffraction audio from this experiment. A few of these explanations are outlined below.

4.4.1 Nonlinear Signal Too Weak?

It is possible that the nonlinear effect is simply too weak to be observed. If so, this would imply that the sound that were were able to detect is from an unexpected source, which may be described by one of the other explanations given below. We now attempt to estimate the strength of the nonlinear sound field:

Suppose our pressure of our system takes the form

$$p(\mathbf{r},t) = (1Pa)\cos(\mathbf{k}_2\cdot\mathbf{r}-\omega_1 t) + (1Pa)\cos(\mathbf{k}_2\cdot\mathbf{r}-\omega_2 t) + Q\cos(\mathbf{k}_d\cdot\mathbf{r}-\omega_d t) \tag{4.20}$$

(This assumes a primary signal strength of 1 Pa and is otherwise equivalent to stating that only the primary signals and the difference signal are relevant - we can ignore the sum signal and higher-order mixing). Then

$$\begin{aligned} p^2 = (1Pa)^2(\cos^2(\omega_1 t)+\cos^2(\omega_2 t)+2\cos(\omega_1 t)\cos(\omega_2 t))\cdot\cdot \\ \cdots + (1Pa)Q\cos(\omega_d t)(2\cos(\omega_1 t)+2\cos(\omega_2 t)) + Q^2\cos^2(\omega_d t) \end{aligned} \tag{4.21}$$

where we have temporarily dropped the spatial dependence of p^2 since only time derivatives of p^2 appear in the equation. Again taking advantage of the trigonometric identitiy $2\cos(\theta)\cos(\phi) = \cos(\theta+\phi)+\cos(\theta-\phi)$, we note that the only term corresponding to an oscillation at the difference frequency is the cross term for the two primary frequencies. If we keep only terms at the difference frequency, then

$$p = Q\cos(\mathbf{k_d}\cdot\mathbf{r}-\omega_d t) \text{ and } p^2 = (1Pa)^2\cos(\mathbf{k}_d\cdot\mathbf{r}-\omega_d t) \tag{4.22}$$

We can plug these values into the Westervelt equation to get an approximate value for Q, the pressure of the difference frequency wave. Noting that the first two terms cancel since $\omega_d = v_s|\mathbf{k_d}|$,

$$Q\delta\omega_d^3\sin(\mathbf{k}_d\cdot\mathbf{r}-\omega_d t) = \frac{\beta\omega_d}{\rho_0 v_s}\cos(\mathbf{k}_d\cdot\mathbf{r}-\omega_d t) \tag{4.23}$$

Ignoring the fact that we're equating a sine to a cosine (we could easily assume a phase shift between the difference frequency and the primary signals that would account for this), we find

$$Q = (1Pa)^2\frac{\beta}{\delta\omega_d^2\rho_0 v_s} \tag{4.24}$$

We assume that the sound diffusivity of air is approximately equal to the thermal diffusivity of air; plugging in values gives

$$Q \approx \frac{(3.81\frac{\text{Pa}}{\text{s}^2})}{f_d^2} \tag{4.25}$$

For a frequency of 1 kHz, $Q = 3.81\times10^{-6}Pa$, and for a frequency of 10 kHz, this becomes $3.81\times10^{-8}Pa$. These signals are significantly quieter than measured above. For instance, in the test described above, we measured an amplitude of over 15 dB for a 10 kHz difference frequency, or over 1×10^{-4} Pa. The calculation above predicts an amplitude of under 4×10^{-8} Pa; at this frequency the measured and calculated amplitudes vary by over three orders of magnitude. Moreover, the measured amplitude from this same test does not appear to scale with $1/f_d^2$ as expected, as the maximum measured amplitude for a difference frequency of 18.75 kHz is actually slightly larger than the maximum measured amplitude at 10 kHz.

There are a few possible explanations for the disagreement between this calculation and our measured results, which are listed below:

- The calculation above is admittedly rather rough, and it is entirely possible that it is simply inaccurate. Ideally, one should fully solve the Westervelt equation in order to determine the amplitude of the focused sound.
- One or more of the explanations given below is responsible for generating a sound field at the difference frequency that is both stronger and wider than the sound field

which we predicted. This unexpected sound field is either generated in lieu of the expected subdiffraction field or is generated alongside this expected field; in either case, the subdiffraction field is at most a weak perturbation to the field that is generated due to the explanations given below.

4.4.2 Difference Frequency Spreading

In the above discussion of nonlinear sound generation, we have assumed that, when the two primary signals mix to generate sound at the difference frequency at a given point, the sound spreads entirely, or at least primarily, along the shared direction of propagation of the two primary signals. This assumption, while reasonable, may not necessarily be true. It remains possible that, when generated, the difference signal spreads in all directions equally, or that, while the signal mostlly propagates in the expected direction, the portion of the signal that spreads in other directions cannot be neglected. See Figure 4.7 for an illustration of the two scenarios. It is of particular interest for our application whether or not the sound generated spreads within the plane of the beam waist of the primary signals. If the sound cannot spread within this plane, then the intensity pattern within the focal plane should have the form assumed above, and the sound field produced in this way could be sub-diffraction limited. However, if the sound does spread within this plane, then the intensity pattern in this plane will be considerably larger than the beam waist of the primary signals due to sound propagating within the plane, such that the sound field will likely not be sub-diffraction limited. Spreading within the focal plane could also allow for interference effects between the difference signals generated at different points in the plane, which could explain the results of the test above. Additionally, a wider spreading pattern would allow for a larger portion of the mixing region to contribute to the intensity of sound generated at a given point, which should increase the expected amplitude, though likely not by multiple orders of magnitude. While this argument is admittedly rather heuristic, the consequences of the spreading pattern are relevant enough to warrant a more rigorous investigation of the issue.

4.4.3 Mixing Along the Beam Length

Although the nonlinear production of the difference frequency is expected to be strongest at the beam waist of the primary signals, this does not mean that nonlinear mixing does not occur at other points in the system. Both primary Gaussian beams are generated from essentially the same area and are focused down to the same point, so the two beams coincide along their entire length. It would not be unreasonable to expect that nonlinear mixing

FIGURE 4.7: A diagram illustrating different possible spreading patterns for the difference frequency signal. The two primary signals (red and blue) travel in a given direction, and nonlinear frequency mixing occurs at a point, generating the difference frequency (purple). On the left, the difference frequency signal travels in essentially the same direction as the two primary signals, and in particular the sound does not travel within the plane normal to the direction of travel. On the right, the difference frequency spreads in nearly every direction.

occurs along the entire length of the primary beams, and that, while we would still expect more sound generation at the beam waist than at a point along the beam length, the contribution of the sound produced along the entire length of the beam to the sound field in the beam waist plane is not negligible. This would also explain the results of the test above - the sound from along the beam would spread out before reaching the waist plane, with the amount of spreading determined by the wavelength of the difference frequency, so the spot size in the beam waist would scale with the wavelength of the difference frequency, and it is possible for the sound from along the beam to cause destructive interference with the sound generated in the waist plane. If this is true, one would also expect a difference signal to be measured along the axis of the system, and although I do not have extensive data on this, I do recall that the sound was present along the axis.

4.4.4 Nonlinear Mixing in the Film

The nonlinear production of the difference frequency may not even be confined to the region of air through which both waves travel, as it is possible that this mixing also occurs within the DLA film itself. This could occur if there is a region of the film for which temperature fluctuations due to the two primary signals are of comparable magnitude. Since DLA films have poor thermal conductivity, one would expect that this only occurs if there is a region of the film that is illuminated by light that is amplitude-modulated at two different frequencies, i.e. if the light from two LEDs overlaps. For DLA films deposited on a thin glass slide, one can place the DLA film close enough to the LEDs that one can safely assume that this effect does not contribute meaningfully to the measured sound field; however, this does pose a possible problem for DLA films deposited on thicker substrates, such as the large mirror blank described above, as the thickness of the substrate prevents one from placing the film close enough to the LEDs to prevent them from overlapping.

In order to eliminate any overlap in the regions illuminated by the LEDs, we designed and 3D-printed an array of lenses. These lenses were to be placed in front of the LEDs to collimate the light from the LEDs such that the areas illuminated by them did not overlap.

However, the lens array did not offer a significant improvement in the measured sound field. We suspect that the lens array failed to collimate the LEDs; the light from a single LED did not seem to focus to a single point, but rather seemed to split into several different points. This could be due to a flaw in the production method - for instance, it is likely that

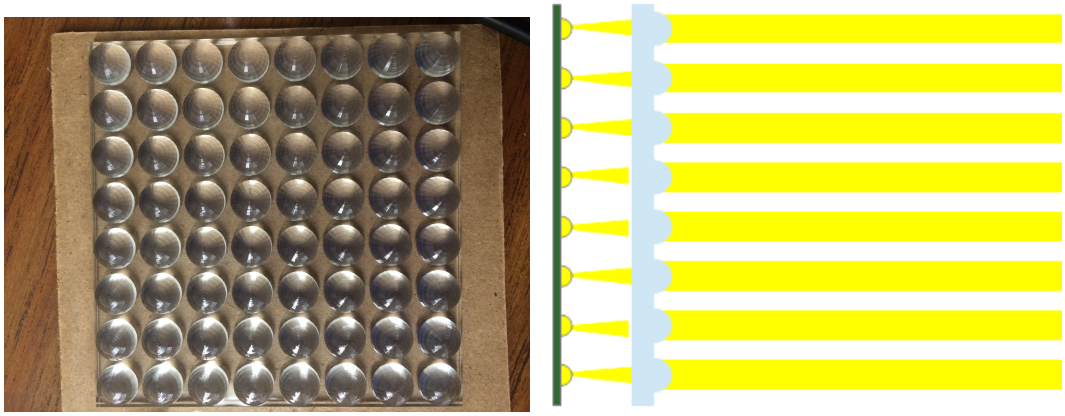

FIGURE 4.8: Left: Top-down view of the lens array. Right: A diagram illustrating how the lens array is intended to focus the light from the LEDs. The light from the LEDs initially spreads, similar to a point source. The light from each of the LEDs passes through one of the "lenses" in the array, and is collimated by the lens.

the individual lenses consist of flat layers of material rather than a curved surface, although we had requested that the lenses in the array be curved by postprocessing.

4.5 Conclusion

We have confirmed that, when one produces sound at two different frequencies in air, sound is generated at the difference frequency. However, although we had expected this sound to be subdiffraction-limited, we have not observed a subdiffraction sound field. We have tried making a number of changes to our system in order to generate a subdiffraction sound field, such as collimating the light from the LEDs with a lens array, using a number of different patterns for the LEDs, and using both planar and curved DLA films; however, these efforts either resulted in no measurable change or no measureable difference frequency field. We have considered a number of different explanations for why the sound field is not subdiffraction-limited, including the possibility of nonlinear generation throughout the system and a significantly-lower-than-expected amplitude for the difference frequency signal; testing the veracity of these hypothesis or taking measures to eliminate the problems suggested by them would appear to be a fruitful avenue for furture research.

Chapter 5

Other and Future Goals

The projects discussed in the previous chapters have been the primary focus of our research with DLA films; however, there are a number of other directions for research into these films that could easily warrant further attention. This chapter serves to outline these other directions of research. One or two of these ideas, such as the evaporation of polymers, have actually been attempted, and our efforts thus far will be described; however, many of these ideas have not been pursued, and what is provided here could be viewed more or less as a series of research proposals for anyone interested in continuing this work.

5.1 Polymer Evaporation

5.1.1 Goal

One of the main issues that prevents these films from being used in many practical applications is that, although the films produce strong signals compared to other photoacoustic sources, the sound production is still relatively weak compared to more conventional transducers. One may suspect that this is in part due to the fact that the photoacoustic signal is generated primarily by density oscillations in the air as opposed to density oscillations in the materials themselves; the thermal expansion of the film is expected to be small compared to the thermal expansion of the air despite the higher temperature oscillation within the film. There are a number of sources [15] [10] [16] that claim that photoacoustic sources that incorporate polymers offer higher efficiencies than purely metallic sources due to the large thermal expansion coefficients of polymers. If one could make a DLA film which undergoes significant thermal expansion, one would expect the photoacoustic response to increase as a result.

Since we suppose that the thermal properties of an individual grain of a DLA film do not differ significantly from the properties of the bulk material, we suspect that a material that undergoes significant thermal expansion in bulk would produce a DLA film with grains that likewise undergo significant thermal expansion. Since polymers typically have large thermal expansion coefficients, a DLA film made of polymers would likely have a strong photoacoustic response, and for this reason we have made some efforts to evaporate polymers.

5.1.2 Efforts Made/Results

Polyethylene was chosen as the first polymer from which we would attempt to make a diffusion-limited aggregate. Polyethylene was chosen largely due to reports that thin films of polyethylene had been deposited by thermal evaporation [17] [18]. which addressed concerns that the evaporation would either not work due to a high boiling point or decomposition of the polymer, or present any sort of safety/health hazard. Details of the various

evaporation efforts can be found in Appendix A; an outline of the parameters varied and results is as follows:

- A molybdenum boat was used for all evaporations. This is primarily because when the other readily-available evaporative source, a tungsten basket, is used, the polyethylene simply slides through a hole in the basket before evaporation occurs.
- When a molybdenum boat is used and a current of around 93 A is run through the boat, evaporation does occur. However, the resulting film appears white, often with some small spots of what appears to be polyethylene that has melted and resolidified. As one might expect of a white object, which does not exhibit strong optical absorption, no significant photoacoustic response is observed.
- Evaporations have been performed at 2 Torr, 10 Torr, 24 Torr, and 210 Torr. Films produced at 2 Torr up to 24 Torr appear about the same. Two evaporations were performed at 210 Torr, and neither resulted in any observed deposition.
- 75 A for about 1.5 to 2 minutes at a pressure of 10 Torr appears to be insufficient to evaporate polyethylene. At 2 Torr this current is sufficient to cause polyethylene to appear to bubble - a surface becomes visible over the edge of the boat, then seems to pop and recede repeatedly. This bubbling may be responsible for the small blobs of solid polyethylene on samples.
- The residue in the boat from a polyethylene evaporation appears dark grey rather than white. Thinking that this change of coloration was due to extended heating of liquid polyethylene and that this change of coloration before evaporation might result in grey or black films, I have tried allowing the current to stay below the actual point at which evaporation occurs for about 10 minutes before actually performing the evaporation. This has not significantly altered the results.

Polyethylene has also been deposited on an old bismuth DLA film to see if the photoacoustic signal of bismuth could be enhanced, but the film's response appeared to be weakened by the deposition.

SEM images of the deposited polyethylene films reveal a structure that differs significantly from the typical DLA structure. The structure of a polyethylene film is much more coarse, featuring much larger grains. Additionally, the structure looks less random than that of a normal DLA film, seeming to feature mostly linear chains of grains rather than having a branchlike structure. This suggests that the polyethylene does not undergo diffusion-limited aggregation, or at least fails to do so at small scales, during evaporation.

If further evaporations of polyethylene are attempted, it would be worth trying to break down polyethylene into smaller polymers before evaporation, perhaps by some chemical reaction. Reducing the polymer size would likely have the effect of reducing the size of the grains in the evaporated film, which could help yield a true DLA structure. Otherwise, evaporation of other polymers could prove fruitful; polycarbonate would be another readily-available polymer worth testing.

5.2 Germanium Nanophotonics

The Schuller lab has requested that we provide them with a germanium DLA film deposited on a germanium wafer. The stated goal of this collaboration is to determine whether germanium DLA films are transparent to photons with energies smaller than the bandgap of bulk germanium, and if so, if the nanostructure causes the DLA films to act as a strong

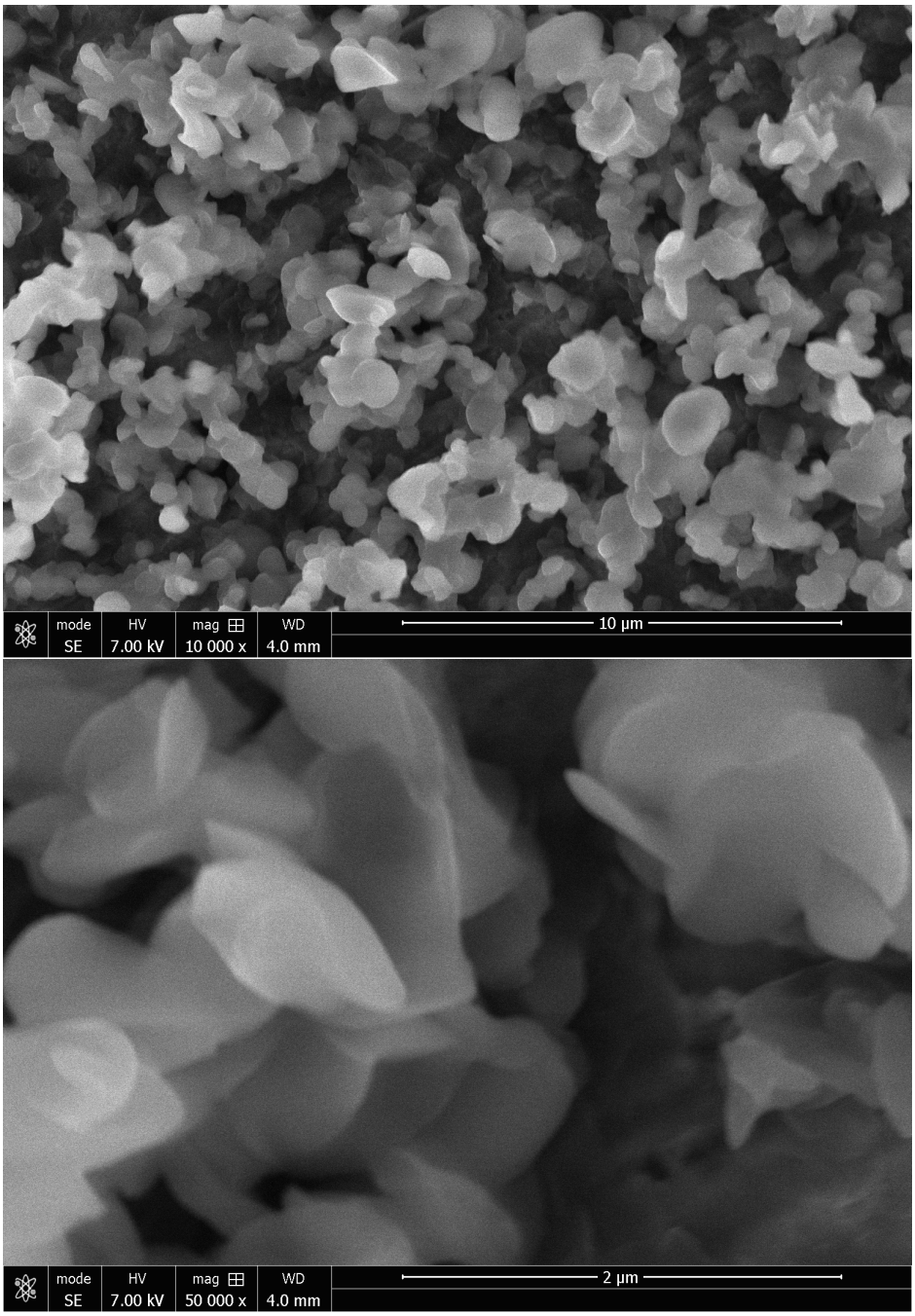

FIGURE 5.1: Two SEM images of an evaporated polyethylene film at different magnifications.

scattering surface for wavelengths that are not absorbed. One such sample was provided to the Schuller lab, and although data was collected on this sample, analysis of this data was hindered by our inability to accurately report the thickness of the film. The Schuller lab requested a thinner sample, and that a portion of the substrate be left clean to aid in measuring the thickness of the film. Such a sample has been provided, and results from the Schuller group are still pending.

5.3 Possible Future Ideas

5.3.1 Silver-Zinc Battery

Batteries which take advantage of nanostructured materials could potentially offer a number of benefits over more conventional batteries; for instance, the higher surface area of a nanostructured electrode, when compared to a bulk electrode, would allow for a greater reaction rate, and thus would increase the power that could be delivered by a battery [19]. In general, it remains possible that the costs of having to produce the required nanostructured materials would outweigh the benefits provided by using these materials; however, since DLA films are both easy and cheap to produce, even a slight increase in performance would outweigh these costs, so long as the DLA structure would not be significantly damaged during operation. With this in mind, it would perhaps be worthwhile to attempt to construct a battery using DLA electrodes and otherwise copying the chemistry of an existing battery. Since we have previously made DLA films of silver and could likely make DLA films of zinc easily (such an evaporation has not been attempted), a silver-zinc battery seems like a reasonable candidate for a first attempt at a DLA battery.

If this project is attempted, there are a few concerns that should be kept in mind when developing the procedures used for this experiment. In order to construct a silver-zinc rechargeable battery, the zinc electrode must be oxidized. When used as a battery, the oxidized film will deoxidize and the unoxidized film will be oxidized. This involves the transfer of oxygen atoms from one film to the other, which could be problematic for a material with a complex structure such as a DLA film. Removal of an oxygen atom from certain sites could possibly disrupt the structure of the DLA film, for instance. While it remains possible that this concern is unjustified, one might recommend forming DLA films of pure zinc and silver, and then allowing one of the films to oxidize, rather than forming a DLA film from an oxide. This would ensure that the structure of the DLA film is not dependent on the presence of oxygen, as the oxygen was added after the structure was formed.

Even if this is done, the main concern with the performance of a DLA battery would be the structural integrity of the DLA film electrodes. Battery operation necessarily requires that charged particles flow from one electrode to the other. It is unclear whether the structure of the DLA films is sturdy enough to remain intact under the pressure provided by this constant particle flow, or if grains or small branches of the structure would be detached from the elecctrodes during operation. Despite these possible issues, this could still be worth experimentation.

5.3.2 Confirmation of Eli Wolf's Earlier Experiment: EM Radiation from Temperature Oscillation

When I first joined the lab, Eli Wolf was working on a project that attempted to generate electromagnetic waves by rapidly heating and cooling a magnetic DLA film around its Curie temperature. Although this project has not been pursued following Eli's graduation, this project may be worth further investigation. Readers who are curious about this project should read chapter 4 of Eli's thesis, although the project can be roughly outlined as follows:

A magnetic material, such as iron or nickel, has an inherent magnetic moment, such that a magnetic field is established both inside and outside the material. However, if a magnetic material is heated above a certain temperature, known as the Curie temperature, it loses this magnetic moment, and there is no net magnetic field due to the material. Now suppose that a magnetic material undergoes a temperature oscillation about its Curie temperature; then the material repeatedly loses and regains its magnetic moment at the frequency of oscillation, in a phenomena referred to as Curie point switching. This generates an oscillating

magnetic field, which one might expect to manifest as electromagnetic radiation. If this process does produce electromagnetic radiation, then one could create a localized optically-actuated radio-frequency source using a DLA film and light amplitude-modulated at the proper frequency.

5.3.3 Structure Factor and Absorption

Original idea from Dr. Vinothan Manoharan, during a discussion about structural color. The intensity of light that is scattered from a material is related to the structure factor of the material in question. The structure factor can be calculated by considering the Fourier transform of the radial distribution function, which gives the density of particles a given distance from a particular particle. More specifically, the structure factor $S((q)$ is given by

$$S(\mathbf{q}) = 1 + n \int g(\mathbf{r}) e^{i\mathbf{k}\cdot\mathbf{r}} \tag{5.1}$$

where $g(r)$ is the radial distribution function [20], which is expected to take the form of a power law for a diffusion-limited aggregate [1]. We should be able to calculate the radial distribution function from existing SEM data, and from this calculate the structure factor and scattering intensity. Although this would not automatically yield the absorption spectrum, since transmitted light still needs to be considered, this would likely still help us understand why DLA films are consistently black.

5.3.4 Polarization in Nonlinear Light Mixing

The idea of using nonlinear phenomena to surpass the diffraction limit is evidently inspired by a technique commonly used in optics, which is assumed to also work for sound since light and sound typically behave analogously. However, the analogy between sound and light is not perfect; for instance, an electromagnetic wave traveling in a particular direction still has a degree of freedom in its polarization, whereas a sound wave traveling in a particular direction is well-defined and the quantity that most directly resembles polarization, the displacement of air particles by the sound wave, is fixed along the direction of propagation. Moreover, some sources indicate that the polarization of light plays an important role in the nonlinear mixing of light; if this is true, then it is entirely possible that the technique used in optics would be ineffective in acoustics. On a related note, some sources also indicate that the direction of propagation of sound plays an important role in the nonlinear mixing of sound [21], and this is likely worth investigation on its own.

5.3.5 Negative Pressure Pulses

Original idea from Dr. Eric Heller.

Our phased array is capable of producing nearly arbitrary sound waves, but it is not without its limitations. One such limitation is that it creates sound in air strictly by decreasing the density of the air, creating a positive pressure pulse in which the pressure is locally increased by the wave's travel. However, pressure waves can generally involve not only positive pressure pulses, but negative pressure pulses as well. Because our system can only cause a decrease in air density relative to the equilibrium state, it may only be possible to generate a pressure greater than the equilibrium pressure, whereas a negative pressure pulse would require the pressure to decrease from its equilibrium value. If our system is in fact incapable of generating a negative pressure pulse, then the ability to reproduce arbitrary sounds would be limited; one example provided by Dr. Heller is that without negative pressure waves, one would not be able to recreate the sound of a clarinet.

One possible solution to this problem of how to generate a negative pressure pulse seems to present itself within the data shown in Figure 2.2. When the frequency of amplitude modulation is small, the system is able to thermalize more quickly than the light switches from on to off, and the microphone measures positive and negative pressure pulses. We have thus already observed negative pressure pulses, which were generated when the system was allowed to approach thermal equilibrium when illuminated and the light was then turned off. However, it is not immediately clear how one would implement this method of generating negative pressure pulses practically. If one simply wishes to generate a brief negative pressure pulse, the problem is rather simple: one can just set all of the LEDs to be on rather than off before generating the signal. For a longer signal, or a more complex signal involving many frequency components, one may not be able to allow for the system to reach an elevated equilibrium by leaving the LEDs on briefly. Perhaps varying the duty cycle of the LEDs for positive and negative pressure pulses would work, but this would also reduce the signal [4].

5.3.6 Psychoacoustic Experiment

Original idea from Dr. Eric Heller.

Suppose we configured our system to produce a continuous pure tone, but to have the source of that tone move randomly and quickly - i.e. a random LED is picked to be the source for a period of time significantly less than one period, another random LED is picked for the next short period, etc. Would one hear a coherent sound? In other words, if one's ear detects a signal that is identical in every way to a continuous tone except that the source is constantly and randomly changing, which manifests as a slight change in the direction of propagation, will the brain interpret that signal as a continuous tone or acknowledge the change in source position? One would suspect that this depends on the angular separation of the sources, with the brain more easily able to detect a change in source position when the two positions have a larger angular separation; this experiment could then determine the angular resolution of one's ability to locate the source of sound.

The programming required for this to work seems rather simple. The relative phases of the LEDs are the same as if the phased array were producing a focus at a point, the point in question being the subject's ear. One simply needs to find a way to have only one LED on at any given time and change which LED is on as a function of time. There are, however, some possible complications. For instance, suppose one wishes, as we typically do, to use a square wave of illumination to produce sound at frequency f. We expect that the frequency components of the produced sound correspond in some way to the frequency components of the illumination, such that the strongest frequency component of the light is the strongest frequency component of the sound. The strongest frequency component of a square wave with frequency f is the component corresponding to frequency f, so using a square wave of a given frequency will produce sound at that frequency. Now suppose that we instead use a square pulse of illumination; this is the illumination pattern that would be used if a portion of a square wave with a temporal width less than one period were used. The Fourier transform of a square pulse of width a is given by

$$\mathrm{sinc}(\frac{\pi}{2a}) = \frac{2a \sin \frac{\pi}{2a}}{\pi} \tag{5.2}$$

If the width a corresponds to half a period os a square wave at the desired frequency, $a = \frac{1}{2f}$, then the Fourier transform will at least have a local maximum at the desired frequency. However, if the square pulse is any shorter than this, then the Fourier transform need not be strong, or even nonzero at this frequency, and in either case many other frequencies will be generated along with the desired frequency. The experiment outlined above calls

for a pure tone that is coherently produced from multiple sources at different times, but if the amount of time an individual source is active is less than the period of the desired frequency, then, according to the discussion above, sound at the desired frequency may not be produced, and many other signals will be generated, such that a coherent pure tone would not be generated. This problem could be avoided by either increasing the switching times to be comparable to or greater than one period or by using a different illumination pattern.

5.3.7 Bending Sound with Heat

Both light and sound deviate from a propagation along a straight path when exposed to a temperature gradient along their path; this phenomena is responsible for the formation of mirages such as fata morgana. Our films can hold a detailed heat pattern due to their low thermal diffusivity and strong optical absorption. Would we be able to further increase our control over the sound field by introducing a sort of steering element that uses DLA films to take advantage of this phenomena? The details of what this would entail have yet to be worked out; I'm currently imagining a ring of glass with a film deposited on the inside, which we then surround with a controllable light source. Heating the film in different positions would change how the sound propagates; for a simple example, heating the film exclusively on one side of the ring would cause sound that was traveling in a straight line to bend away from the heated side. It is probable that the bending would only be very slight, or that only the portion traveling very near the edge of such a device would be effectively bent; regardless, this idea may warrant an attempt.

5.3.8 Damage Mechanism

The first germanium film that we produced began to show signs of damage in its center a couple of days after it was produced. This is despite the fact that this film was never used with our phased array. The only possible explanation I can think of is that the film's damage was caused by the flash on my phone's camera, as I took a picture of the film shortly after producing it; but no signs of damage were immediately evident after the picture. See Figure 2.4 for images of this germanium film before and after the damage became optically apparent and SEMs of damaged and undamaged portions of the film. The mechanism by which the films begin to exhibit damage may be worth further investigation; in particular, I would be interested in taking a picture of a germanium film then recording footage of the film in an SEM to watch how the structure evolves over the day or two it takes for visible damage to the film to become apparent, or optically damaging a film while simultaneously viewing the film in an SEM, although SEM time may be too restricted to easily carry out this experiment.

5.3.9 Stereo/Other psychoacoustics

We have shown that one can produce intensity fields of sound corresponding to two foci at different locations with ultrasound; however, we have not explicitly verified that, if these foci were in the audible range, a person would actually perceive the sound at two different points. It would be a simple matter to make two foci with one at each ear and characterize the sound field produced by ear.

5.3.10 Underwater Photoacoustics

Although this would limit the immediate applicability of some of our results, there are a number of benefits to measuring photoacoustic signals in water as opposed to in air. The

sound signals produced would likely be much stronger, the nonlinear mixing of sound is also a stronger effect in water than in air, and hydrophones tend to be cheaper than microphones. The only concerns would be the possibility that the films are damaged in water or that the photoacoustic effect somehow fails underwater (perhaps due to the hydrophobicity of some of these films), and both of these concerns could be easily tested without purchasing any new equipment.

Appendix A

Evaporation Notes

A.1 Bismuth

General method: To deposit a bismuth film onto a glass slide, copper film, or other such small substrate, a small amount of bismuth should be evaporated in a molybdenum boat. A current of 115 amperes through this boat for about 4.5 minutes should be sufficient to form a film well-suited to photoacoustic applications. For larger substrates that are placed on the platform above the boat, a longer evaporation using more material or multiple evaporations may be required to reach the desired thickness; typically two evaporations with the same parameters as used for a normal evaporation are used, and this produces a thin but effectice DLA film on the substrate. For more details on the evaporation process, see Chapter 1, under the section "More Detailed Process".

TABLE A.1: Bismuth Evaporation, May 11, 2015

Flushes: 79.5 mTorr, 54.8, 45.1, 38.8, 34.5 $\rightarrow$ 2.04 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	initial jump to 40, but then 20.27	2.04
1:00	41.2	2.04
1:30	61.2	2.04
2:00	78.2	2.06
2:30	91.1	2.06
3:00	115.2	2.04
3:30	115.7	2.04
4:00	115.1	2.06
4:30	115.3	2.04
5:00	114.4	2.06
5:30	116.1	2.06
6:00	115.6	2.04
6:30	115.8	2.06
7:00	115.9$\downarrow$ 0	2.06

Results: The bismuth in the copper holder came out pretty well.

TABLE A.2: Bismuth Evaporation, June 25, 2015

Flushes: 76.7 mTorr, 51.4, 41.6, 36.3, 32.0 ↑ 1.98 open

Time	Current (Amps)	Pressure(Torr)
0:30	20.64	1.98
1:00	39.8	1.98
1:30	60.4	1.98
2:00	79.5	1.98
2:30	99,5	1.98
3:00	116.9	1.98
3:30	117.0	1.98
4:00	117.6	1.98
4:30	117.7	1.98
5:00	115.8	1.98
5:30	115.3	1.98
6:00	115.8	1.98
6:30	116.0	1.98
7:00	116.3	1.98

Results: The film in the copper hanger was terrible. The one in the holder was decent on one end but slightly melted on the other. Trying again with the holder farther + at steeper angle.

TABLE A.3: Bismuth Evaporation, June 26, 2015

Flushes: 53.8 mTorr, 41.2, 34.4, 30.4, 27.3 ↑ 1.98 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	20.57	1.98
1:00	40.5	1.98
1:30	60.5	1.98
2:00	81.7	1.98
2:30	99.8	1.98
3:00	114.8	1.98
3:30	116.0	1.98
4:00	116.0	1.98
4:30	116.1	1.98
5:00	116.7	1.98
5:30	116.6	1.98
6:00	116.5	1.98
6:30	116.9	1.98
7:00	[blank]	[blank]

[Results: although the results weren't explicitly noted on this evaporation, a comment on the next implies that this and other immediately preceding evaporations were terrible]

TABLE A.4: Bismuth Evaporation, June 29, 2015 (1 of 2)

Pellets: 4 (well, 2 + leftover, if I recall, correctly)
Flushes: 58.9 mTorr, 44.8, 34.5, 29.8, 26.5 ↑ 2.00 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	26.53	2.00
1:00	40.7	2.00
1:30	60.5	2.00
2:00	80.5	2.00
2:30	99.2	2.00
3:00	115.4	2.00
3:30	115.8	2.00
4:00	115.8	2.00
4:30	116.0	2.00
5:00	115.2	2.00
5:30	115.5	2.00
6:00	115.8	2.00
6:30	115.9	2.00
7:00	115.2	2.00

Results:
Hanger: okay; melted at top (or is it just less materials?)
Mirror: That failed

TABLE A.5: Bismuth Evaporation, June 29, 2015 (2 of 2)

Pellets: large remainder +2
Flushes: 50.5 mTorr, 39.8, 33.8, 29.8, 27.2 ↑ 2.00 Torr closed (possible leak)

Time	Current (Amps)	Pressure(Torr)
0:30	Jumpy(18-25)	2.00
1:00	40.3	2.00
1:30	61.2	2.00
2:00	81.6	2.02
2:30	101.3	2.02
3:00	115.6	2.04
3:30	115.2	2.06
4:00	115.1	2.08
4:30	115.3	2.08
5:00	115.2	2.10
5:30	115.0	2.12
6:00	115.2	2.12
6:30	115.5	2.14
7:00	115.6	2.16

Note: Pressure dropped to 2.14 when voltage was being lowered
Results: Possibly useable! (sp) (only problem is corner by screw)

TABLE A.6: Bismuth Evaporation, June 30, 2015 (1 of 3)

Pellets: 2 + large residue
Flushes: 47.6 mTorr, 39.7, 34.5, 31.3, 26.0 ↑ 1.96 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	19.20	1.96	
1:00	40.3	1.96	
1:30	59.2	1.98	
2:00	80.6	1.98	
2:30	110.9	2.00	highest A[mps]: 118 (brief)
3:00	115.0	2.02	
3:30	115.1	2.02	
4:00	115.2	2.04	
4:30	115.7	2.06	
5:00	115.7	2.06	
5:30	116.1	2.08	
6:00	116.6	2.10	
6:30	114.8	2.12	
7:00	115.5	2.14	Changed to 2.12 while lowering A

Results: Seems like it would have been a good sample, if not for some scars. Maybe I messed it up while removing or maybe cleanliness of slide is a variable?
Note: Kurt said Krutik and Eli spent a lot of time cleaning the chamber. Maybe that's it?

TABLE A.7: Bismuth Evaporation, June 30, 2015 (2 of 3)

Pellets:1 because large residue
Flushes: 47.0 mTorr, 36.0, 31.3, 28.1, 26.0 ↑ 2.02 Torr closed

Time	Current (Amps)	Pressure(Torr)
0:30	20.18	2.02
1:00	41.0	2.02
1:30	61.4	2.02
2:00	80.3	2.04
2:30	101.1	2.04
3:00	115.0	2.06
3:30	115.1	2.08
4:00	115.2	2.08
4:30	115.2	2.10
5:00	115.3	2.12
5:30	115.5	2.12
6:00	115.4	2.14
6:30	115.6	2.16
7:00	115.8	2.18

Results: Excellent (at least as good as yesterday's usable one)
Update: I now see a fingerprint on it, hopefully still good.

TABLE A.8: Bismuth Evaporation, June 30, 2015 (3 of 3)

Pellets: 1 (but I don't even think it needs it)
Flushes: 46.6 mTorr, 37.0, 32.7, 29.1, 26.5 ↑ 2.02 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	20.06	2.04	
1:00	41.0	2.04	[Current] went high for a bit
1:30	59.5	2.04	
2:00	79.6	2.04	
2:30	101.0	2.06	
3:00	115.2	2.08	
3:30	115.2	2.10	
4:00	115.2	2.10	
4:30	115.3	2.12	
5:00	115.4	2.14	
5:30	115.4	2.14	
6:00	115.5	2.16	
6:30	115.4	2.18	
7:00	115.8	2.18	

Results: Film looks decent, but with a few marks on it. Possibly still usable, but I'll try to make more

TABLE A.9: Bismuth Evaporation, July 1, 2015

1 new pellet
Flushes: 45.4 mTorr, 35.6, 30.4, 27.2, 25.4 ↑ 2.00 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	20.16	2.00	
1:00	40.7	2.02	
1:30	59.5	2.02	went too high for a bit...66ish
2:00	81.4	2.02	
2:30	100.3	2.04	
3:00	115.5	2.08	
3:30	115.5	2.08	
4:00	115.7	2.08	
4:30	115.8	2.10	
5:00	115.8	2.10	
5:30	116.1	2.12	
6:00	116.0	2.14	
6:30	116.1	2.14	
7:00	116.1	2.16	

Results: About as good as the last ones; one big mark, otherwise good. Should be usable, especially if mark is between LEDs or on a dead one.

TABLE A.10: Bismuth Evaporation, Undated (likely July 1, 2015)

1 new pellet
Flushes: 45.1 mTorr, 36.5, 31.3, 26.0, 23.8 ↑ 1.94 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	20.92	1.96	
1:00	38.4	1.96	[Current] high (45) for a bit
1:30	60.7	1.96	
2:00	83.0	1.96	
2:30	100.1	1.98	
3:00	114.5	2.00	
3:30	115.6	2.00	
4:00	115.7	2.02	
4:30	115.4	2.02	
5:00	115.3	2.04	
5:30	115.1	2.06	
6:00	115.0	2.06	
6:30	115.4	2.08	
7:00	115.5	2.08	

Results: I don't see any marks, looks good to me... but still one more.

TABLE A.11: Bismuth Evaporation, Undated

Pellets: 2 + residue. Flushes: 78.0 mTorr, 57.6, 45.2, 38.0, 24.7 ↑ 2.00 Torr closed
Note: I cleaned out the evaporator before this run

Time	Current (Amps)	Pressure(Torr)
0:30	19.63	2.00
1:00	39.2	2.00
1:30	60.8	2.02
2:00	79.2	2.02
2:30	99.7	2.04
3:00	115.8	2.04
3:30	116.1	2.06
4:00	115.1	2.08
4:30	115.4	2.10
5:00	115.8	2.10
5:30	116.0	2.12
6:00	116.6	2.14
6:30	115.7	2.16
7:00	115.9	2.18

Results: Thinnest on the bottom...

TABLE A.12: Bismuth Evaporation, Undated

Pellets: 0 (Residue) Flushes: 50.5 mTorr, 37.8, 31.7, 28.1, 25.7 ↑ 1.98 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	21.02	1.98
1:00	41.3	1.98
1:30	58.5	1.98
2:00	80.4	1.98
2:30	100.0	2.00
3:00	115.4	2.00
3:30	116.0	2.00
4:00	116.4	2.00
4:30	115.5	1.98
5:00	115.9	2.00
5:30	116.0	2.00
6:00	116.0	2.00
6:30	116.4	2.00
7:00	116.7	2.00

Results: Very thin, residue still there.

TABLE A.13: Bismuth Evaporation, Undated

Pellets: 0 (residue) Flushes: 57.5 mTorr, 41.1, 33.8, 29.2, 26.2 ↑ 2.00 Torr closed

Time	Current (Amps)	Pressure(Torr)
0:30	22.01	2.02
1:00	40.8	2.02
1:30	61.2	2.02
2:00	80.4	2.04
2:30	100.6	2.04
3:00	115.1	2.06
3:30	115.1	2.08
4:00	115.2	2.08
4:30	115.2	2.10
5:00	115.4	2.12
5:30	115.3	2.12
6:00	115.4	2.14
6:30	115.6	2.16
7:00	115.9	2.18

Results: Thin on bottom, residue still there...Let's run the next evap longer. I'll try open again, at least it was uniformly bad

TABLE A.14: Bismuth Evaporation, Undated

Pellets: 0 (Residue), Flushes: 44.1 mTorr, 35.1,30.4, 27.4, 25.6 ↑ 2.06 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	18.81	2.06
1:00	40.1	2.06
1:30	62.3	2.06
2:00	80.9	2.06
2:30	100.4	2.06
3:00	117.1	2.06
3:30	117.8	2.06
4:00	118.0	2.06
4:30	117.3	2.06
5:00	117.6	2.06
5:30	117.9	2.06
6:00	117.9	2.06
6:30	117.1	2.06
7:00	117.2	2.06
7:30	117.2	2.06

Results: I can tell some of the residue evaporated, but the film was still bad.

TABLE A.15: Bismuth Evaporation, Undated

Flushes: 37.0, 29.7, 26.6, 24.8, 23.0 ↑ 1.86 closed

Time	Current (Amps)	Pressure(Torr)
0:30	20.31	1.88
1:00	42.1	1.88
1:30	60.6	1.88
2:00	81.6	1.90
2:30	100.2	1.90
3:00	115.2	1.92
3:30	115.2	1.94
4:00	115.2	1.94
4:30	115.5	1.96
5:00	115.7	1.96
5:30	115.6	1.98
6:00	115.9	1.98
6:30	116.1	2.00
7:00	116.3	2.02

Results: A step in the right direction, but still seems a bit thinner near edges. Passable though, let's repeat.

TABLE A.16: Bismuth Evaporation, Undated

No new pellets. Flushes: 51.7, 41.5, 34.8, 30.8, 28.2 ↑ 1.84 closed

Time	Current (Amps)	Pressure(Torr)
0:30	19.23	1.86
1:00	40.7 (but went high for a while)	1.86
1:30	61.6	1.86
2:00	80.6	1.88
2:30	99.6	1.88
3:00	114.9	1.90
3:30	116.2	1.92
4:00	116.7	1.94
4:30	116.9	1.94
5:00	116.8	1.96
5:30	116.9	1.98
6:00	117.0	2.00
6:30	117.1	2.02
7:00	116.8	2.02

Results: good enough.

TABLE A.17: Bismuth Evaporation, Undated

No new pellets. Flushes: 52.2, 42.5, 36.0, 32.2, 29.6 ↑ 1.90 closed

Time	Current (Amps)	Pressure(Torr)	
Notes 0:30	20.24	1.92	
1:00	391.7	1.92	
1:30	60.4	1.92	
2:00	79.9	1.92	
2:30	102.4	1.94	
3:00	115.1	1.96	
3:30	115.8	1.98	
4:00	115.9	1.98	smell burning
4:30	116.0	2.00	
5:00	115.9	2.02	
5:30	115.8	2.04	
6:00	115.8	2.06	
6:30	115.7	2.08	
7:00	115.6	2.10	

Results: Really thin on the bottom. I'm tempted to change the boat, but I'll try one more open first

TABLE A.18: Bismuth Evaporation,Undated

New boat, 3 pellets, noticed something was up with sample hanger (could tilt up and down)
Flushes: 50.2, 36.1, 32.0, 29.4, 27.5 ↑ 1.92 open

Time	Current (Amps)	Pressure(Torr)
0:30	20.60	1.92
1:00	39.8	1.94
1:30	69.5	1.94
2:00	79.8	1.94
2:30	102.6	1.94
3:00	114.7	1.94
3:30	115.7	1.94
4:00	116.4	1.94
4:30	116.3	1.94
5:00	116.5	1.94
5:30	116.3	1.94
6:00	116.4	1.94
6:30	116.5	1.94
7:00	116.3	1.94

Results: Melted to the point of being unusable on top. Try moving sample holder back.

TABLE A.19: Bismuth Evaporation, Undated

Residue + 2 pellets, Flushes: 46.7, 38.1, 28.7, 26.4, 25.1 ↑ 1.94 open

Time	Current (Amps)	Pressure(Torr)
0:30	19.93	1.94
1:00	41.5	1.94
1:30	65.9 (wild)	1.94
2:00	78.9	1.94
2:30	100.1	1.94
3:00	115.1 (went up to 119)	1.94
3:30	115.0	1.94
4:00	115.0	1.94
4:30	115.8	1.94
5:00	115.6	1.94
5:30	115.6	1.94
6:00	115.5	1.94
6:30	115.5	1.94
7:00	115.2	1.92

Results: Thin or melted on top, I can't actually tell which this time.

TABLE A.20: Bismuth Evaporation, Undated

Residue + 2 pellets, Flushes: 80.9, 57.4, 32.7, 29.4, 26.9 ↑ 1.82 closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	21.88	1.82	
1:00	42.3	1.82	
1:30	58.8	1.82	big jump
2:00	80.4	1.84	
2:30	102.2	1.84	
3:00	115.2	1.86	
3:30	115.0	1.88	
4:00	115.3	1.88	
4:30	115.4	1.90	
5:00	115.6	1.92	
5:30	115.5	1.94	
6:00	115.5	1.94	
6:30	115.5	1.96	
7:00	115.5	1.98	

Results: This. I like this. Let's repeat this three times.

TABLE A.21: Bismuth Evaporation, Undated

Residue + 2 pellets, Flushes: 62.0,46.8,38.5,33.6, 30.0 ↑ 1.82 closed

Time	Current (Amps)	Pressure(Torr)
0:30	19.91	1.84
1:00	42.5	1.84
1:30	61.2	1.84
2:00	82.9	1.86
2:30	100.3	1.86
3:00	115.3	1.88
3:30	115.1	1.90
4:00	115.0	1.92
4:30	115.1	1.92
5:00	115.4	1.94
5:30	114.8	1.96
6:00	115.0	1.98
6:30	115.5	1.98
7:00	115.5	2.00

Results: Not perfect, but comparable to 1st. 2 more.

TABLE A.22: Bismuth Evaporation, Undated

Residue + 2 (1 small), Flushes:65.0, 42.7, 33.6, 28.7, 26.0 ↑ 1.82 closed

Time	Current (Amps)	Pressure(Torr)
0:30	20.66	1.84
1:00	42.0	1.84
1:30	61.3	1.84
2:00	78.0	1.84
2:30	101.6	1.86
3:00	116.2	1.88
3:30	115.4	1.88
4:00	115.3	1.90
4:30	115.4	1.90
5:00	115.1	1.92
5:30	115.0	1.94
6:00	115.4	1.96
6:30	114.9	1.96
7:00	115.5	1.98

Results: A little [scarred], and thin at one corner, otherwise good.

TABLE A.23: Bismuth Evaporation, Undated

Residue + 2 pellets, Flushes: 56.1, 40.0, 32.8, 28.7, 26.3 ↑ closed [blank]

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	19.91	1.82	
1:00	40.2	1.82	
1:30	60.4	1.84	
2:00	79.9	1.84	
2:30	101.3	1.86	
3:00	114.8	1.88	
3:30	116.9	1.88	
4:00	116.9	1.88	
4:30	116.5	1.90	
5:00	116.5	1.92	hear buzzing
5:30	116.2	1.92	
6:00	116.0	1.94	
6:30	116.0	1.94	
7:00	116.2	1.96	

Results: Peculiar. Almost cloudy when back-illuminated. Although now that I look at the third one again, so is it. I think they're still usable, so I'll wait before making more for now.

TABLE A.24: Bismuth Evaporation, July 30, 2015

Residue + 1 pellet, Flushes: 68.5, 48.6, 38.7, 35.6, 31.7 ↑ 1.92 closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	17.56	1.92	
1:00	41.9	1.92	
1:30	61.9	1.94	
2:00	80.0	1.94	
2:30	100.7	1.96	
3:00	115.6	1.98	
3:30	115.8	2.00	
4:00	115.9	2.00	
4:30	115.0	2.02	
5:00	114.8	2.04	
5:30	114.5	2.06	Recorded late
6:00	114.8	2.08	
6:30	114.7	2.10	
7:00	114.8	2.12	

Results: Thin, with odd markings on the bottom. They do not appear transparent, except at their boundaries. My first though is of some organic material, but I cannot say why.

TABLE A.25: Bismuth Evaporation, Undated

Residue +2 pellets, Flushes: 49.1, 37.9, 32.8, 29.7, 27.3 ↑ 1.82 closed

Time	Current (Amps)	Pressure(Torr)
0:30	19.81	1.82
1:00	41.2	1.82
1:30	62.3	1.82
2:00	79.8	1.82
2:30	99.1	1.84
3:00	117.1	1.86
3:30	116.5	1.86
4:00	116.4	1.88
4:30	116.2	1.88
5:00	116.2	1.90
5:30	116.2	1.92
6:00	116.0	1.92
6:30	115.8	1.92
7:00	115.7	1.96

Results: Basically inaudible from back. Looks OK in front though

TABLE A.26: Bismuth Lens + glass Evaporation, Undated

Residue + 2, Flushes: 57.7, 40.9, 34.7, 31.4, 29.0 ↑ 1.78 closed

Time	Current (Amps)	Pressure(Torr)
0:30	24.04	1.78
1:00	39.8	1.78
1:30	61.9	1.78
2:00	81.6	1.80
2:30	102.7	1.80
3:00	115.1	1.82
3:30	116.8	1.82
4:00	116.3	1.84
4:30	116.1	1.86
5:00	116.1	1.86
5:30	115.8	1.88
6:00	115.5	1.90
6:30	115.4	1.90
7:00	115.1	1.92

Results: The glass came out better than the lens, but both are okay.

TABLE A.27: Big Lens Evaporation Attempt 1

4 pellets, Flushes: 65.7, 32.6, 29.6, 27.5, 25.7 ↑ closed 1.96 1.98

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	26.06	1.98	
1:00	43.3	1.98	huge spike initially
1:30	60.0	1.98	
2:00	78.3	2.00	
2:30	99.9	2.02	
3:00	118.5	2.04	
3:30	120.9	2.06	
4:00	117.9	2.06	
4:30	119.3	2.08	
5:00	118.6	2.08	
5:30	119.4	2.10	
6:00	118.6*2.10	might have misread; <120	
6:30	119.4	2.12	
7:00	117.9	2.14	

Results: way too thin, can't even tell if uniform or not.

TABLE A.28: Big Lens Evaporation Attempt 2 (part 1)

Large residue + 5 pellets, Flushes: 63.4, 53.8, 27.8, 25.8, 24.2 ↑ 1.48 closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	23.81	1.46	
1:00	41.9	1.46	
1:30	60.4	1.48	
2:00	78.6	1.48	
2:30	94.7	1.50	
3:00	115.5	1.52	
3:30	116.3	1.52	
4:00	116.9	1.52	
4:30	116.7	1.54	
5:00	117.3	1.54	
5:30	117.9	1.56	
6:00	118.4	1.56	
6:30	114.1 or 119.1	1.58	late
7:00	119.7	1.58	
7:30	120.3	1.60	
8:00	120.2	1.60	
8:30	120.4	1.62	
9:00	120.7	1.62	
9:30	121.0	1.64	
10:00	121.5	1.66	
10:30	122.5	1.66	
11:00	122.7	1.68	
11:30	123.3	1.70	
12:00	123.6	1.72	

Results: Pretty good, but weak on one side. The boat's placement is slightly asymmetric; I'll have to fix that later, but for now let's rotate 180 degrees and try again.

TABLE A.29: Big Lens Evaporation Attempt 2 (part 2)

smaller - but still large - residue + 5 pellets. Flushes: 52.4, 39.0, 33.0, 28.8, 26.3 ↑ 1.49 closed

Time	Current (Amps)	Pressure(Torr)
0:30	22.66	1.44
1:00	40.9	1.46
1:30	61.1	1.46
2:00	80.5	1.46
2:30	103.2	1.48
3:00	117.4	1.50
3:30	117.8	1.50
4:00	118.7	1.50
4:30	119.2	1.52
5:00	119.5	1.52
5:30	119.8	1.54
6:00	120.2	1.54
6:30	117.4	1.54
7:00	117.0	1.56
7:30	117.3	1.56
8:00	117.5	1.58
8:30	117.7	1.58
9:00	118.1	1.58-1.60
9:30	118.1	1.60
10:00	118.8	1.60
10:30	118.9	1.62
11:00	119.2	1.62
11:30	119.6	1.64
12:00	119.9	1.66

[Although results were not explicitly provided, the results were good enough to be used, although it was not used because the lens was slightly damaged following this evaporation]

TABLE A.30: Big Lens Evaporation 3

Residue + 5 pellets, Flushes: 48.8, 39.9, 34.0, 30.2, 27.2 ↑ 1.48 closed

Time	Current (Amps)	Pressure(Torr)
0:30	20.68	1.48
1:00	41.2	1.48
1:30	59.6	1.50
2:00	82.2	1.50
2:30	102.5	1.52
3:00	116.8	1.54
3:30	117.5	1.54
4:00	118.6	1.54
4:30	119.2	1.56
5:00	119.6	1.56
5:30	119.8	1.58
6:00	119.5	1.58
6:30	119.4	1.60
7:00	119.9	1.60
7:30	120.0	1.62
8:00	120.2	1.64
8:30	120.2	1.64
9:00	120.6	1.66
9:30	120.8	1.68
10:00	121.1	1.68
10:30	121.4	1.70
11:00	121.4	1.72
11:30	121.9	1.74
12:00	121.8	1.74

Results: A little thinner than the last film, but still good. Reevaporating (flipped)

TABLE A.31: Big Lens Evaporation 3 part 2

Residue + 4, Flushes: 54.1, 38.0, 31.7, 28.0, 25.2 ↑ 1.46 closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	22.8	1.46	
1:00	41.6	1.46	
1:30	59.4	1.96	
2:00	81.5	1.48	
2:30	101.6	1.48	
3:00	115.5	1.50	
3:30	116.0	1.52	
4:00	116.4	1.52	
4:30	116.7	1.52	
5:00	117.2	1.54	
5:30	117.5	1.54	
6:00	117.5	1.54	
6:30	117.6	1.56	
7:00	117.7	1.56	
7:30	117.7	1.58	Late
8:00	117.7	1.58	
8:30	117.8	1.60	
9:00	117.7	1.60	
9:30	117.8	1.62	
10:00	117.8	1.62	
10:30	117.9	1.64	
11:00	118.2	1.64	
11:30	118.1	1.66	
12:00	118.3	1.66	

[Results were again not stated, but the film was good enough to use]

TABLE A.32: Big Lens Reevaporation

Residue +4 pellets, Flushes: 63.7, 47.6, 39.0, 34.7, 32.0 ↑ 1.78 closed

Time	Current (Amps)	Pressure(Torr)
0:30	19.99	1.80
1:00	41.7	1.80
1:30	59.1	1.80
2:00	80.1	1.82
2:30	81.3?	1.82
3:00	116.0	1.84
3:30	116.1	1.86
4:00	117.0	1.88
4:30	117.1	1.88
5:00	117.5	1.90
5:30	117.7	1.90
6:00	118.2	1.92
6:30	116.9	1.94
7:00	117.2	1.94
7:30	117.5	1.96
8:00	118.1	1.98
8:30	118.5	1.98-2.00
9:00	116.3	2.00
9:30	116.5	2.02
10:00	116.5	2.04
10:30	116.3	2.06
11:00	116.1	2.08
11:30	116.2	2.08
12:00	116.5	2.10

Results: Light but about what was expected. Now for second half.

TABLE A.33: Big Lens Reevaporation Part 2

Residue + 4 or 5 pellets. Flushed: 57.8, 44.1, 38.0, 33.6, 31.0↑ 1.70 closed

Time	Current (Amps)	Pressure(Torr)
0:30	23	1.70
1:00	42	1.70
1:30	63	1.70
2:00	81.6	1.72
2:30	100	1.74
3:00	115.3	1.76
3:30	115.7	1.76
4:00	116.0	1.76
4:30	116.1	1.78
5:00	116.3	1.78
5:30	116.5	1.80
6:00	116.5	1.80
6:30	116.6	1.80
7:00	116.4	1.82
7:30	116.4	1.82
8:00	116.5	1.84
8:30	116.7	1.84
9:00	116.5	1.86
9:30	116.7	1.86-1.88
10:00	117.4	1.88
10:30	117.7	1.90
11:00	117.9	1.90
11:30	118.1	1.92
12:00	118.2	1.94

Results: Still light, but responds reasonably well

TABLE A.34: Bismuth Evaporation for large lens, Undated

Evaporation 1
Residue + 2 pellets, Flushes: 51.3 mTorr, 40.6, 34.5, 31.8, 29.9 ↑ 1.60 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:15	23.08	1.60	
0:36	41.8	1.60	
1:15	71.8	1.62	went faster than expected
1:41	87.0	66.2	
2:06	106.1	1.62	1.64 2.20
2:40	115.0	1.64	
3:00	120.0	1.66	
3:30	119.4	1.66	
4:00	120	1.66	Maybe go lower?
4:22	122.5	1.68	
5:00	122.9	1.68	going back to 115
5:30	117.4	1.68	
5:44	118.8	1.20	self-rising
6:30	118.0	1.70	
7:00	116.7	1.70-2	
8:10	118.3	1.72	just changed to 1.74
9:00	120.0	1.74	just to 1.76
10:00	121.0	1.76	
10:15	121.1	1.78	
11:00	121.5	1.78	
11:15	121.3	1.80	
12:30	123	1.82	
12.55	122.4	1.84	stopping soon
13:30	122.0	1.84	just to 1.86; stopping
14:35	0	1.84	Done

TABLE A.35: Bismuth Evaporation for large lens, Undated (part 2)

Evaporation 2
Residue + many pellets, Flushes: 51.8 mTorr, 40.5, 35.8, 32.6, 30.8 ↑ 1.58 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:05	20.94	1.60	
0:37	41.8	1.60	
1:10	64.7	1.60	
1:40	80.4	1.60	
2.05	101.6	1.62	
2:40	114.7	1.62	Falling fast
3:00	118.0	1.64	
3:30	117.2	1.64	Thinking of increasing
4:00	118.5	1.64	dropped to 116.7
4:20	125.4	1.66	
5:00	124.1	1.66	
5:25	123.4	1.68	
6:00	123.8	1.68	
6:35	117.4	1.68	I decreased it
6:53	118.5	1.68-70	
7:30	118.6	1.70	
8:17	118.4	1.72	
9:15	118.6	1.72	
9:45	124.2	1.74	Increased it again
10:30	123.0	1.74	this one seems slow
11:40	122.9	1.76	
11:50	122.6	1.78	
11:35*124.0	1.80		
13:30	123.4	1.82	at 1.84 I stop
14:23	122.5	1.84	stopping
15.15	-	1.82	done

*This is the time written in my notes, but is likely a typo
Results: good enough.

A.2 Nickel

TABLE A.36: Nickel Evaporation Attempt 1: April 28, 2015

"Four pieces of nickel were used in the wire boat". [Note: "the wire boat" refers to the W basket]
Flushes: 46.5 mTorr, 35.8, 30.4, 27.0, 25.0 ↑ 2.02 Torr open

Time	Current (Amps)	Pressure(Torr)
0:30	20.6	2.02
1:00	46.6	2.02
1:30	60.0	2.04
2:00	81.4	2.04
2:30	110→ 0	2.04

Comments:
"Voltage dropping fast"
"How Did This Work?!?" [Context: The evaporation ended rather quickly with the basket breaking. Despite this, a nickel DLA film was successfully formed.]

TABLE A.37: Nickel Evaporation 2: April 29, 2015

Flushes: 58.8 mTorr, 43.8, 35.1, 30.1, 27.0 ↑ 2.04 Torr

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	11.55	2.04	
1:00	20.7	2.04	
1:30	29.7	2.04	
2:00	37ish	2.04	
2:30	50ish	2.04	Glow starts
3:00	62	2.04	
3:30	62	2.04	
4:00	62	2.04	
4:30	71	2.04	
5:00	71	2.04	
5:30	82	2.04	
6:00	87	2.04	
6:30	70	2.04	
7:00	69	2.04	
7:30	69.2	2.04	
8:00	69	2.04	
8:30	68	2.04	
9:00	67ish	2.04	
9:30	67.7	2.04	
10:00	[blank, presumably 0]	2.04	

A.3 Germanium

TABLE A.38: Germanium Evaporation Attempt 1

Pellets: residue 4
Flushes: 43.4 mTorr, 35.6 $\pm$ 0.3ish, 32.1, 29.7 $\pm$ 0.1, 28.4 $\uparrow$ 1.82 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
1:00	20	1.84	Trying to keep 20A. Trends down
2:00	35	1.84	Tried to maintain 40A. will try again
2:30	36	1.84	Already glowing, no signs of evaporation
3:30	39.4	1.86	Still no signs of evap. Boat glows strongest near top. Current rising to 40.5, then dropping again
5:00	43.3	1.88	Current changing without input. The brightness gradient is drastic
6:00	45$\pm$2	1.90	Big oscillations (43$\rightarrow$45)
6:55	50	1.90	Just increased current to 50
7:25	49-53	1.92	More oscillations
8:00	51.4	1.92	other rings now glowing, top ring and bottom connector brightest
9:00	59.8	1.96	current increased
10:00	59.9	1.98	Now bright enough that I can see sample holder fell onto the basket. Increasing pressure suggests the film is being formed. abort evap
12:30	0	2.06	Evap aborted

Results: Target that fell on the basket showed some rainbow-like discoloration and the germanium melted. It's unclear if any evaporated.

TABLE A.39: Germanium Evaporation Attempt 2

Pellets: residue +2
Flushes: 46.4 mTorr, 37.8 $\pm$ 0.3, 33.1, 30.3, 28.4 $\uparrow$ 1.96 [Torr] closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:12	19.76	1.96	Just started
1:00	30.01	1.96	Up to 30. No glowing yet
1:30	26.82	1.96	Declining current on its own
1:55	37	1.98	Just increased. Amps falling fastish
2:25	35.1	1.98	faint glowing starts
3:00	37.1	2.00	Still falling, top ring clearly glows
4:30	40.7	2.02	Increased current a little while ago
5:00	40.5	2.02	Brighter glowing but no signs of evap
5:30	49.7	2.02	This time the top connector glows more.
			I find the brightness concerning
7:00	55.2	2.06	Note to self - check if boat is deformed later
8:00	59.7	2.08	How is no liquid Ge falling out the bottom?
8:40	80.3	2.12	Now the bottom is glowing.
			Also, I don't recall turning the knob after 60...
9:30	80.8	2.16	The connectors glow more than the actual basket.
			Still no evidence of evap
10:40	81.5	2.24	Pressure rising fast. That may count as evap evidence
11:35	81.4	2.32	
12:12	81.6	2.36	
12:24	81.6	2.38	
13:10	81.7	2.44	Still no visible evidence of evap.
			We have extra boats, so let's go a bit higher
14:12	85.3	2.54	
14:35	85.4	2.58	Actually, let's kill the evap and see how the pressure changes
16:45	0	2.66	Finished dropping current. Pressure climb slowed
			but continued,slight drop at the end.
17:35	0	2.66	Pressure still high. Maintenance at high pressure suggests
			minimal cooling and minimal removal of Ge particles from air.
			Does Ge not adhere to Cu? Also not just a leak
18:15	0	2.64	
19:37	0	2.62	
20	0	2.62	Filling with argon now

Results: An interesting success. A copper film hung overhead barely had any film on it and was only slightly discolored, but still has a not-that-weak response. A veritically-hung piece of copper was kind of a greyish-black and had a stronger responce, and the film deposited on a coper film placed on the bottom of the chamber angled towards the basket had a portion as black as we are used to, with a PA signal similar to bismuth. There is also a large black spot immediately below the boat.

TABLE A.40: Germanium Evaporation Attempt 3

Pellets: residue +2

Flushes: 30.8 mTorr, 27.9, 26.5, 25.4, 25.1 ↑ 1.90 [Torr] closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:11	13.10	1.90	Let's use 10A increments
0:30	12.71	1.92	
1:06	20.72	1.92	(↑20A)
1:30	19.41	1.92	(Amps dropping)
1:53	29.52	1.92	(↑30A)
2:30	28.33	1.92	No drastic pressure rise yet
2:53	41.4	1.92	↑40A
3:15	42.2	1.94	Amps self-rising a bit
3:53	49.7	1.94	↑50A. Faint glow starts
4:22	49.5	1.94	glow from bot connector
4:45	55.1	1.94	↑55A
5:05	53.9	1.96	
5:36	61.7	1.96	↑60A, glow strong
6:22	60.0	1.96	Amps stable, no pull up to 80
7:05	67.1	1.98	(I increased it)
7:37	66.9	1.98	Basket itself barely glowing
8:07	66.7	2.00	Can't tell if film deposited yet
8:38	70.6	2.00	↑
9:10	70.1	2.00	I think no dep. yet
9:45	74.5	2.02	↑
10:15	75.2	2.04	
10:45	80.0	2.06	↑
11:15	78.5	2.08	I think some has deposited
11:45	78,5	2.08	Connectors radiant, basket is glowing too now
12:10	78.4	2.10	
12:55	79.7	2.12	
13:20	79.6	2.14	The black actually looks pretty nice
13:45	79.7	2.14	Possible burning on a corner, looks grey
14:22	79.6	2.18	Let's give it 2 minutes more
15:00	79.3	2.20	
15:30	74.4	2.22	
16.00	74.3	2.24	Let's stop now
17:30	0	2.22	Done

Results: One of the edges is a little gray, but this is, I think, the blackest film I have seen in ages.

TABLE A.41: Germanium Evaporation, May 9, 2017

Pellets: Amount already there +2 (5 or 6? It is a lot)
Flushes: 105 mTorr, 87.4, 71.0, 65.7, 76.8, 77.1 ↑ open 1.94 torr
Other notes: This evaporation was done to test if depositions could still occur as normal despite the vacumm leaking somewhat.

Time	Current (Amps)	Pressure(Torr)	Notes
0:15	21.75	1.94	
1:05	36.5	1.94	Falling Fast
2:00	39.4	1.94	Can't keep at 40; only now stopped falling (could this be melt point?)
2:35	39.6	1.92-4	Top ring glows
3:20	49.4	1.92	
4.25	59.2	1.92	
5:25	77.2	1.92	Was trying to raise to 70; jumped to 80
5:50	78.6	1.92-4	
7:15	84.4	1.92	
7:30	104??	1.92	Rapid jump
9:00	0	1.92	Started descent 8

Results: The deposited film was black in a small region below the basket and a brown, rust-like color further from the basket. The rust-colored film still has a strong photoacoustic response, comparable to that of the bismuth film currently in use although it's close and may be comparing apples+oranges (copper, new vs. glass, old)

TABLE A.42: Germanium Evaporation, May 10, 2017

Pellets: Residue +2
Flushes: 110 mTorr, 84.2, 76.8, 67.1, 66.0, 63.0 ↑ open 1.90 Torr

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	20.04	1.90	
1:00	36.0	1.90	
2:00	48.9	1.90	Starting to glow
2:55	57.3	1.90	
4:10	66.6	1.90	
5:00	73.7	1.90	
6:00	79.7	1.90	
7:00	83.6	1.90	No jumps thus far
8:30	85.1	1.90	Going to start descent at 9:00
9:00	87.6	1.90-2	Saw jump to 100 after long steady period
11:15	0	1.90	Actually started drop 9:40

Results: Rust again. This time, even the top of the big-lens-mount is coated somewhat.

TABLE A.43: Germanium Evaporation, May 11, 2017

Pellets: Residue +4;
Flushes: 114 mTorr, 89.4, 74.0, 70.8, 69.7, 65 ↑ 2.12 Torr open

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	20.26	2.12	
0:55	38.1	2.14	
1:25	39.5	2.12	No/less drastic dropping
2:15	57.1	2.12	
3:10	61.4	2.12	Slight jump on its own
3:55	69.4	2.12-4	
5:20	79.0	2.14	Watching for jump
7:15	79.4	2.12	Current knob 40, lower than usual
8:20	79.9	2.12	Try 85 again
8:45	82.9	2.14	
9:40	85.0	2.12	
10:50	84.6	2.12	Stopping
11.55	0	2.12	

Results: Still rust. The sudden spike cannot be responsible. Also worthy of note, there is/has been one spot directly below the basket where the deposited film is black. Maybe just place the sample there?

TABLE A.44: Germanium Evaporation, May 12, 2017

Pellets: residue +3-4?
Flushes: 122 mTorr, 92.5, 78.7, 83.9, 67.1, 64.3 ↑ 1.96 [Torr] open

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	21.02	1.96	
1:00	39.0	1.96	
1:50	58.6	1.96	
3:15	69.9	1.96	Copper looks like something on it
4:55	79.4	1.96	Looks to me like deposition
7:10	0	1.96	[Started stopping] at 6:30

Results: Region of black below the basket is larger, but there is still rust-colored film. This might be good enough.

TABLE A.45: Germanium Evaporation, May 15, 2017

Pellets: 4 or 5
Flushes: 114 mTorr, 92.2, 80.1, 71.6, 64.3, 66.6 ↑ 1.98 [Torr] open

Time	Current (Amps)	Pressure(Torr)	Notes
0:07	20.61	1.98	
0:45	34.47	1.98	dropping fast
1:45	37.9	1.98	stopped dropping
2:55	59.5	1.98	glowing, some self-rise in current
3:35	65.8	1.98	rose on its own
4:35	69.5	1.98	yes I rose it this time
5:36	73.6	1.98	pressure sometimes dropping to 1.96
6:40	84.2	1.98	Expect a jump at some point, current's been lagging
7:50	84.4	1.98	No surprises
10:30	85.6	1.98	Jump starting
12:45	75.5	1.98	Current reducing itself. That's weird (or because I tried to stop jump)
14:55	0	1.96	Started descent 13:30

Results: The black region at the bottom of the chamber is larger, and even the rust colored portions look darker Also, no apparent signs of residue (maybe slight). Could the rust color just be the color of a thinner Ge film? (or the new boat helps)

TABLE A.46: Germanium Evaporation, May 16, 2017

Note: evaporating onto a Ge wafer, attempting a thin film.
Pellets: slight residue +1
Flushes: 108 mTorr, 88.5, 71.2, 68.8, 60.6, 63.8↑ 2.12 Torr open

Time	Current (Amps)	Pressure(Torr)	Notes
0:20	22.49	2.12	oscillating a lot, now at 25
1:40	334.0	2.12	
3:30	55.3	2.12	
5:25	69.0	2.12	
6:40	83.4	2.12	
7:50	84.7	2.12	can't see it, but stopping in 30
9:15	0	2.12	Started descent 8:25

Results: It looks kind of rust-colored on the wafer, but still looks black in the chamber (well, with a little red).

A.4 Polyethylene

TABLE A.47: Polyethylene Evaporation Attempt 1 - August 30, 2016

4 pellets. The pellets are of reasonable size
3 targets: 1) Below basket; 2) Propped up beside basket; 3) Overhead and hanging down.
Using a W basket
Flushes: 54.0 mTorr, 40.5, 35.5, 32.3, 28.6 ↑ closed 1.72 Torr

Time	Current (Amps)	Pressure(Torr)	Notes
0:00	10	1.72	started
0:30	20	1.72	
1:00	31.24	1.72	First rise 1:15
1:30	29.6	1.74	A[mps] dropping fast
2:15	35	1.76	A drops fast
2:40	29.5	1.78	A self-dropping, glowing started
3:30	32	1.80	Samll spot on target 1), possible deposition on targets 1+2
5:00	31.5	1.82	I hope it didn't just fall out bottom
6.36	35.38	1.84	The basket looks empty
7:30	35.66	1.84	Okay, I'm pretty sure it just slid out. Stopping
8:46	0	1.84	Done

Only giving it 1/2 hour to cool.

It definitely fell through. Some white deposition below basket. No PA effect observed.

TABLE A.48: Polyethylene Evaporation Attempt 2

Using a Mo boat this time.
Pellets: 3
Flushes: 57.6, 45.1, 37.7, 34.3, 31.5 ↑ 1.82 [Torr] closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	19	1.84	Hang low for first 15 s
1:00	22.3	1.84	
1:15	33.78	1.84	
1:55	45	1.84	
2:30	45.0	1.86	
2:45	54.6	1.86	
3:30	66.2	1.86	
4:00	67.1	1.86	
4:15	81.4	1.86	Rose to 1.88 just after writing
4:50	78.9	1.88	
5:15	92.2	1.90	
5:35	92.2	1.92	
6:10	91.7	1.94	I think it's going now
6:45	90.7	1.96	
7:35	92.3	1.96-8	Alternating
8:00	96.2	1.98	(increased in hopes of faster)
8:30	95.1	1.98	
8:40	94.6	2.00	
10:00	96.3	2.02	
11:00	98.0	2.02	Give it 3 minutes
11:12	98.5	2.04	[current] rising on its own
12:12	99.0	2.04-6	Alternating
13:12	98.7	2.08	Stopping
14:24	[blank]	[blank]	Done

Results: White. Best on Target 2. No PA effect.

TABLE A.49: Polyethylene Evaporation: May 31, 2016 (1 of 2)

This time, trying evaporating onto old bismuth films to see if their PA signal is enhanced.
Pellets: 3
Flushes: 53.1 mTorr, 29.7, 27.9, 26.2, 26.1 ↑ 1.84 [Torr] closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	21.45	1.86	Let's go up 20 A per minute until 90
1:07	40.4	1.86	
2:10	61.1	1.86	
3:00	61.4	1.86	saw some voltage drop, staying here a bit longer
3:40	82.2	1.88	Accidentally went to 90 for a second first
4:30	91.6	1.88	
4:43	91.4	1.90	
4:55	90.7	1.96	Wow that's fast
5:10	89.7	1.92	Voltage self-dropping
5:50	91.8	1.96	
6:10	92.2	1.98	
7:00	91.6	2.00	
8:00	92.0	2.00	It's definitely slowed down
8:25	93.9	2.02	
9:00	93.3	2.02	
10:00	93.1	2.04	Very faint boat glow
11:00	93.9	2.04	
11:50	93.6	2.06	
13:00	97.7	2.06	
13:30	92.8	2.08	
14:00	92.5	2.08	1 more minute then stop
15:00	93.1	2.08	stopping
16:10	0	2.08	stopped

Results: So much white. PA basically gone, front and back. Let's try for a thinner film.

TABLE A.50: Polyethylene Evaporation: May 31, 2016 (2 of 2)

Pellets: 1
Flushes: 57.5 mTorr, 38.5, 33.1, 30.2, 28.5 ↑ 1.86 [Torr]

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	20.22	1.86	Same [current] as last time
1:10	40.2	1.86	
1:50	42.0	1.86	saw it go to 1.88 for a second
2:30	62.3	1.88	
3:10	81.3	1.88	
4:20	92.6	1.90	
4:30	91.5	1.92	
5:05	92.9	1.92-4	
6:00	92.3	1.94	
6:20	92.1	1.96	
7:10	92.5	1.96	
7:55	92.2	1.98	when it hits 2 I['ll] stop it
9:00	91.5	1.98	
9:50	91.9	2.00	stopping
11:00	0	1.98	Done

Results: Still too thick, but can hear signal now.

TABLE A.51: Polyethylene Evaporation, undated

Pellets: 4
Flushes: 52.5 mTorr, 41.8, 35ish, 31.2, 28.7 ↑ 23.0 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:25	24	23	
0:50	41.0	23.0	
1:20	62.3	23.0	
1:45	71	23.5	
2:10	91.9	23.5	
2:30	91.6	24.0	
2:55	89.8	24.5	
3:25	94.1	25.0	
3:45	95.2	24.5	Alt's between 24.5 and 25
4:35	94.9	25.0	stopping by 7
5:00	95.6	25.0	maybe sooner
5:15	95.8	25.0	like now
6:10	0	24.5	Rose to 25.5 as stopping

Results: All of the PE was evaporated, but was white again. No PA. Some more solid droplets (remelted?). Let's try slightly shorter and 200 Torr.

TABLE A.52: Polyethylene Evaporation, undated

Pellets: 4
Flushes: 45.4 mTorr, 36.5, 32.6, 30.1, 28.5 ↑ 220 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	20.84	210	
0:37:40.4	210		
1:05	65	210	also showing 205
1:40	83.0	205	
2:05	93.2	205	Let's give it 3 minutes
2:35	89.2	205-10	
3:50	90.4	205-10	almost stable at 210
5:00	92.1	210	stopping very soon (5:25)
6:05	0	210	done

Results: Where'd the polyethylene go??

TABLE A.53: Polyethylene Evaporation, undated

Pellets: 3
Flushes: 74.2 mTorr, 45.6, 39.1 ↑ 1.96 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	22.2	1.97	
1:00	46.7	1.98	
1:30	62.4	2.00	
2:00	92.9	2.00	stopping
2:30	[blank]	[blank]	
3:00	[blank]	[blank]	

Cool for 15 minutes
Results: PE melted in boat (that was the grey!) but no deposition. I should let the next run go longer at 93.

TABLE A.54: Polyethylene Evaporation, undated

Pellets: residue
Flushes: 59.5, 40.1, 35.1 ↑ 1.90 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:30	21.6	1.92	
1:00	48	1.94	
1:30	64	1.94	
2:00	91	1.91	rapidly dropping
2:30	91	1.96	
3:20	0	2.00	There was a large current spike (102+ around 2:45); I started to stop after that spike

Results: White. The evaporation appears to have completed this time, leaving little to no residue.
Nine small squares of copper were arranged in a square pattern. The two rows that were further from the boat were shadowed by a sample holder placed over them. Deposition occurred on the furthest row; deposition was white and exhibited no PA effect. The middle row had less noticeable evaporation, with the end furthest from the boat being less white. Still no PA. The closest row had the most noticeable deposition with some large white spots that could suggest remelting. Still white, no PA. At least one of the samples in Row 1 felt coarse to touch.

TABLE A.55: Polyethylene Evaporation, undated

Pellets: 3
Flushes: 63.2 mTorr, 47.3, 39.2 ↑ 9.70 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	20	9.80	
0:37	45.4	9.80	
1:07	76.1	9.85	
1.16	74.8	9.90	
1:30	74.5	9.95	Let's see what staying here does
1:55	72.5	9.95-10.0	stopping (2:10)
2:30	0	9.95	Done

Results: The polyethylene does not seem to have evaporated - there is still a large residue in the boat. As the residue seems to contain multiple discrete pieces, I doubt it even melted for long - I would expect a more uniform distribution from a liquid. There is, at most, a light dust on some of the samples, but one such sample (the only one I checked) had no PA, and the dust is light enough that it could easily be nothing and trivial (fingerprints?). Also, the vacuum chamber smells odd.

TABLE A.56: Polyethylene Evaporation, undated

Pellets: Residue
Flushes: 53.6 mTorr, 43.8, 36.0 ↑ 9.85 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:10	24.46	9.85	
0:31	46.4	9.85	9.90 at .49
1:05	73.5	9.90-9.95	10 at 1.22
1.31	88.4	10.0	Rapid drop
2:00	92.5	10.0	Less rapid drop
2:30	91.9	10.5	Nearly stopped drop - stopping
3:25	0	10.0	Done

Results:
Still smells funny
Definitely some white on row 1, maybe some on row 2, almost certainly none on row 3
Central cavity of boat turned grey
The grey can be scratched off, indicating, to me at least, that the grey is the residue.

TABLE A.57: Polyethylene Evaporation, undated

Pellets: 3
Flushes: 48.3 mTorr, 35.1, 31.1 ↑ 10.0 Torr open

Time	Current (Amps)	Pressure(Torr)	Notes
0:15	26	10.0	Dropping fastish to 23
0:53	46.7	10.0	went high for a bit on accident
1:25	71.1	10.0	Dropping again
2:04	90	10.0	Dropping. Pressure to 9.95 2:28
3:00	92.6	10.0	stopping; might have logged wrong pressure
3:43	0	9.90	Done
4:00	0	9.95	weird, it went up?

Results: switched to 4 strips of copper. Closest strip does not appear white, has some large blobs at PE. No PA. The next closest is white with some large blobs. The next closest is pretty much blob-free and still white, while the furthest is a thinner white. None of the strips exhibited PA. Also, noticed small areas with some black, and found a loose unidentified pellet in the chamber.

TABLE A.58: Polyethylene Evaporation, undated

Pellets: 3
Flushes: 55ish mTorr, 43.9, 36.8
Note: Closing it went to 38ish, and there is definitely a small leak. Let's go fastish.

Time	Current (Amps)	Pressure(mTorr)	Notes
0:15	23.5	44.2	
0:43	42.8	45.6	
1:15	68.7	47.3	
1:45	97.9	49.7	
2:00	90.8	51.4	is it just me or is it [...] faster
2:30	90.9	61.8	yes it is. stopping at 2:50
3:30	0	108	done

Results: Somewhat odd. Much less white than before in the chamber, but maybe that's just from cleaning. There is a white ring, but it's pretty thin. As usual, the closest sample just has blobs. 2 and 3 have some white, which makes sense since they are in the ring. Two additional targets, placed under the main bottom of the vacuum chamber, seem unaffected. I do find the presence of a black spot in the center intriguing/encouraging, but the closest sample might suggest that this is just remelt.

TABLE A.59: Polyethylene Evaporation, undated

Pellets: 3
Flushes: 57.6 mTorr, 46.2, 37.9 ↑ 1.68 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:12	22.8	1.68	had trouble starting voltage
0:40	49.4	1.68	
1:10	70.5	1.70	Let's stay here for 10+ minutes
2:52	71.8	1.72	
5:13	70.0	1.74	slight down trend for current
7:23	71.2	1.76	
9:40	70.4	1.78	
11:30	71.2	1.80	Soon (let's say 13:30) we'll go up to 92[A] for 30 seconds
12:45	71.7	1.80	Afterwards, we stop
13:35	91.8	1.84	fast rise, saw 1.86 for a second
14:48	0	1.84	Done

Results: Still white, including samples behind the shadow shield. Also worthy of note, I found two unaltered pellets in the chamber. Retry?

TABLE A.60: Polyethylene Evaporation, September 20, 2016

Pellets: 1 new pellet + 2 old pellets
Flushes: 65.3 mTorr, 49.0, 42.2 ↑ 1.98 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:15	21.06	1.98	2.00 at 0:28
0:40	46.2	2.00	
1:10	77.1	2.00	
1:45	81.0	2.00	2.02 1:55
2:40	79.1	2.02	
3:10	76.4	2.04	This seems [...] stable, so I'll be back later
5:00	78.3	2.06	To clarify, I'm still watching the run, just not paying as much attention
9:00	78.2	2.14	The amps slowly rise
14:20	78.9	2.24	At 15:00 let's turn it up
15:12	90.5	2.28	
15:30	91.8	2.28	off now
16:17	0	2.30	Done

Results: Looks whiter than usual

TABLE A.61: Polyethylene Evaporation, September 22, 2016

Pellets: 3
Flushes: 57.1, 47.5, 39.6 ↑ 1.94 Torr closed

Time	Current (Amps)	Pressure(Torr)	Notes
0:22	27	1.96	
0:47	41.9	1.98	
1:17	67.2	1.98	Huge jump: 40ish →70ish→40ish→80
1:47	70.6	1.98-2.00	
3:50	71.5	2.02	
5:37	69.1	2.04	The system is incredibly jumpy. Will either not change current or change a ton.
8:23	80.2	2.06	
10:30	77.0	2.10	Just changed to 2.12
14:40	75.2	2.18	Stopping now
15:20	0	2.18	Done

TABLE A.62: Polyethylene Evaporation, undated

Pellets: Residue +2
Flushes:33.1 mTorr, 32.1, 29.2↑ 1.90 Torr

Time	Current (Amps)	Pressure(Torr)	Notes
0:20	20.47	1.90	Start by going to 50A
0:45	35.4	1.90	No visible change to 1:30
1:45	47-54	1.90	Big spikes/swings
2:40	54.9	1.92	swings seem to have calmed. No signs of evap
4:00	61.2	1.92	Dropped to 50 while I was looking in chamber
5:45	54.4	1.92	Outer cage + reflected light = hard to see in there
6:20	65.4	1.94	Let's try 65 now
8:10	71.8	1.94	Pulled self up
10:00	70.4	1.96	Still no obvious signs of evap
10:45	78.1	1.96	Higher
12:10	74.6	1.98	Reflectivity of closer glass slide suggests no evap yet
13:40	72.4	2.00	I think I see the PE bubbling. A surface rises over the boat's edge, then seems to pop + recede. Timescale 1 second
17:30	74.5	2.04	Bubbling continues. Give this a few minutes - if it stops, increase amps
20:50	75.4	2.10	
28:00	79.5	2.30	Meter died for a second. Looks like bubbling's stopped and evap may be happening. Stopping at 30
31:00	80.1	2.34	stopping
31:00	0	2.30	done

Results: So white! Except for one side of the glass panel closer to the boat, all targets are quite white (although the copper appears less white than the glass). The absence of white on one end of the substrate is kind of odd, and it coincides well with where I was trying to shine the light - maybe there's a limit, but this is far from conclusive.

TABLE A.63: Polyethylene Evaporation, June 1, 2017

Pellets: 4
Flushes:116mTorr, 95.5, 80.0 ↑ 215 Torr closed
Note: Using a new boat; looks like molybdenum but could be tungsten, just going by shape of boat.
Note: Loud noise when filling with argon; could this be the leak?

Time	Current (Amps)	Pressure(Torr)	Notes
0:15	19.70	210	some self-rise/fluctuation
1:20	33.66	205	
2:35	48.2	205	
4:00	62.1	205	still a lot of self-rise
6:10	80.1	205	no glow yet, but could be harder to see since elevated
6:45	75.8	205	some jumps to 210
7:20	86.4	210	
8:30	92.6	210	
9:50	100.6	210	Brief jumps to 215 earlier, another at 10:30
12:45	109.4	215	
14:15	119.7	215-220	still no glow
15:35	126.5	220	stopping soon
17:10	0	220	stopped

Results: Thin if any deposition, some small spots, no PA, odd smell. I didn't give it forever to cool though.

TABLE A.64: Polyethylene Evaporation, June 2, 2017

Pellets: 4 (no apparent residue)
Flushes:118mTorr, 91.0, 84.5↑ 2.02 Torr open
Goal: Anneal but do not evap PE so it turns grey/black

Time	Current (Amps)	Pressure(Torr)	Notes
0:07	22.30	2.02	
1:10	41.4	2.02	
2:20	53.2	2.04	stay here a while
13:30	53.1	2.04	stopping

Results: Checked June 15, 2017. Residue is grey and smooth on the surface. No deposition.

TABLE A.65: Polyethylene Evaporation, June 15, 2017

Pellets: Residue
Flushes:117mTorr, 90.4, ↑
Note: Loud noise when filling with argon again

Time	Current (Amps)	Pressure(Torr)	Notes
0:07	21.69	2.18	
0:50	41.1	2.18	
1:20	64.3	2.18	
2:15	82.8	2.18	Dropping
3:11	91.1	2.18	stay at 93
5:07	95.2	2.18	
6:05	96.5	2.18	self-rising
8:10	97.1	2.18	
9:30	97.4	2.18	
10:55	97.8	2.18	stopping

Results: Still white, no PA. Preheating seems to have been ineffective.

Appendix B

SEM Data

This appendix contains a representative sample of the SEM data collected of various DLA films, organized by the type of film. A more complete collection of SEM Data can be found on the lab's Dropbox.

B.1 Bismuth

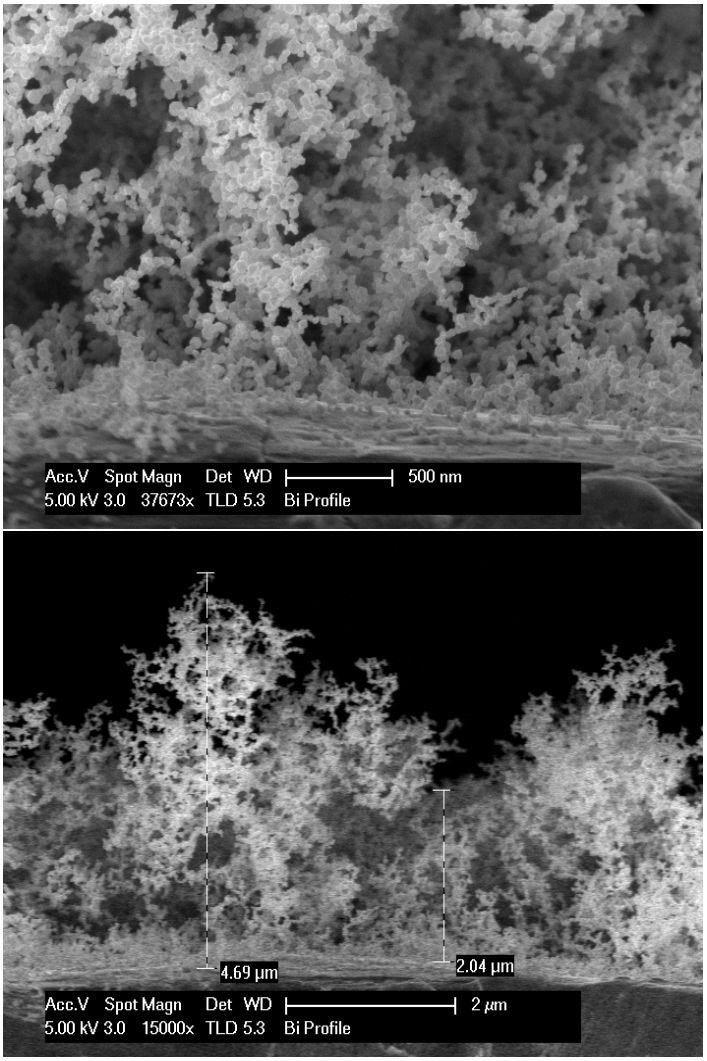

FIGURE B.1: SEM images of a bismuth DLA film from a side view, taken on April 27, 2015

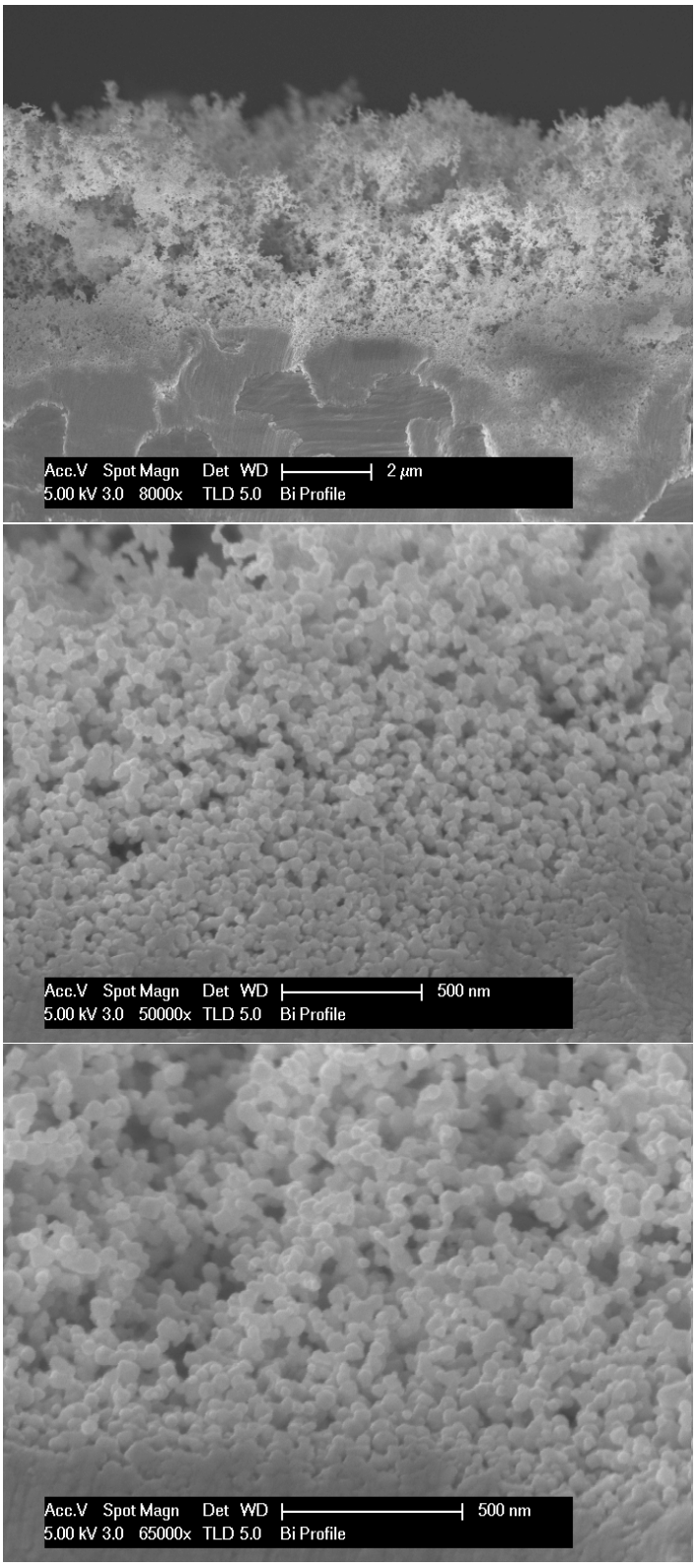

FIGURE B.2: SEM images of a different bismuth DLA film from a side view, taken on April 27, 2015

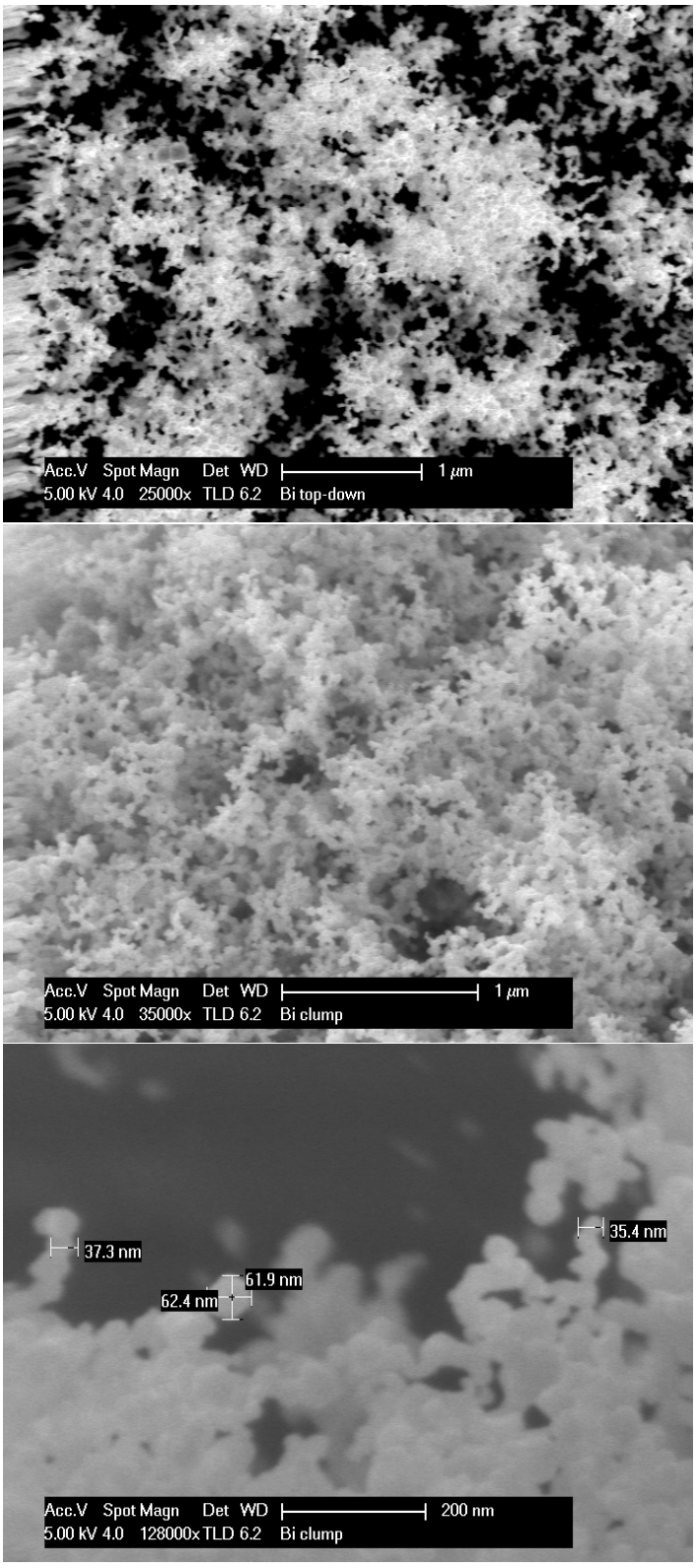

FIGURE B.3: SEM images of a bismuth DLA film from a top-down view, taken on April 27, 2015

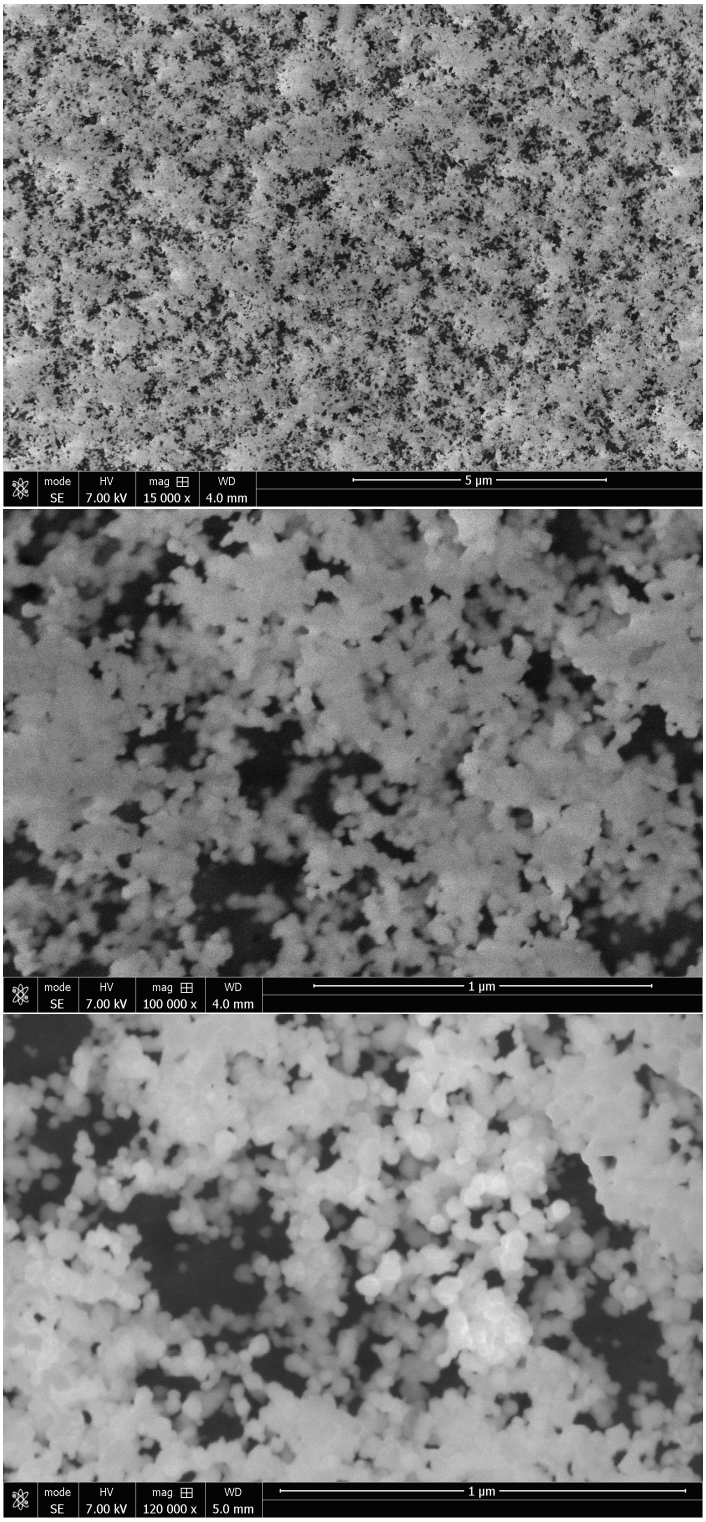

FIGURE B.4: SEM images of a bismuth DLA film from a top-down view, taken on September 12, 2016.

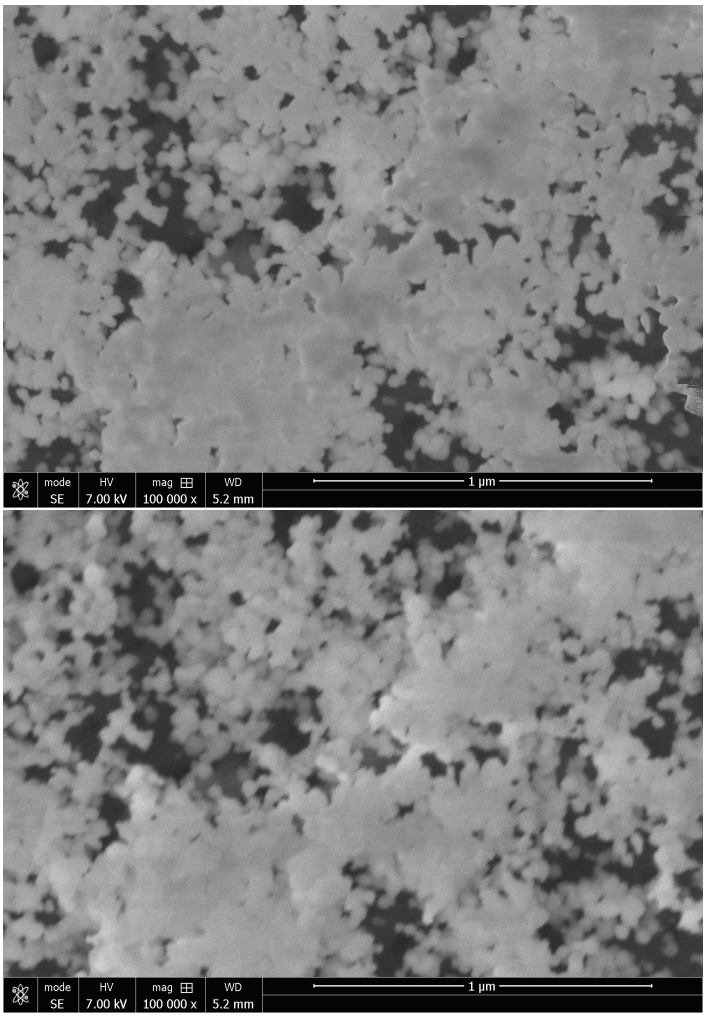

FIGURE B.5: SEM images of a bismuth DLA film that has been damaged by prolonged exposure to amplitude-modulated light from the phased array. The images are of the same sample as that imaged in the previous figure (although the portion imaged above was undamaged), and were also taken on September 12, 2016. More images of this film can be found in the next figure.

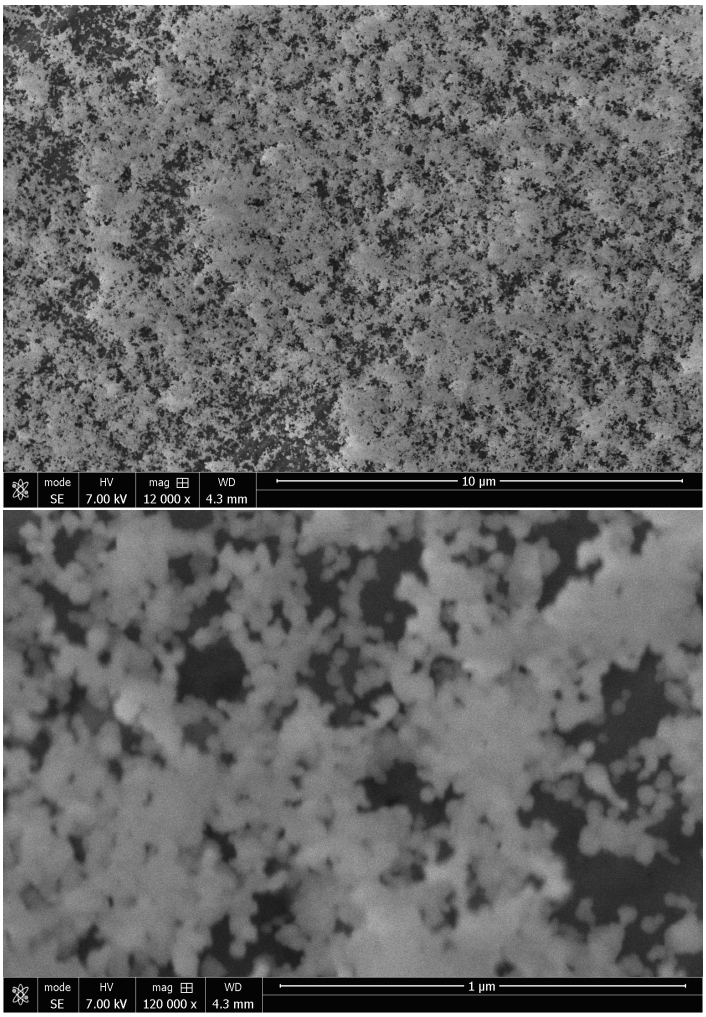

FIGURE B.6: SEM images of a bismuth DLA film that has been damaged by prolonged exposure to amplitude-modulated light from the phased array. The images are of the same sample as that imaged in Figure B.4 (although the portion imaged above was undamaged), and were also taken on September 12, 2016. More images of this film can be found in the previous figure.

B.2 Iron

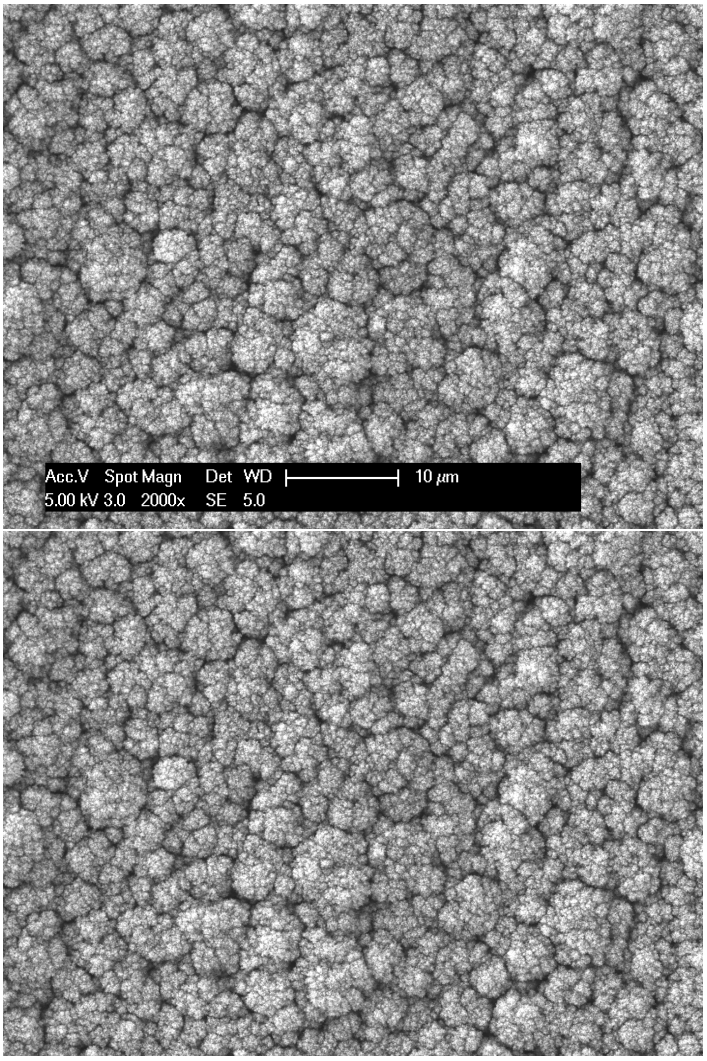

FIGURE B.7: SEM images of an iron DLA film from a top-down view, taken on April 16, 2015

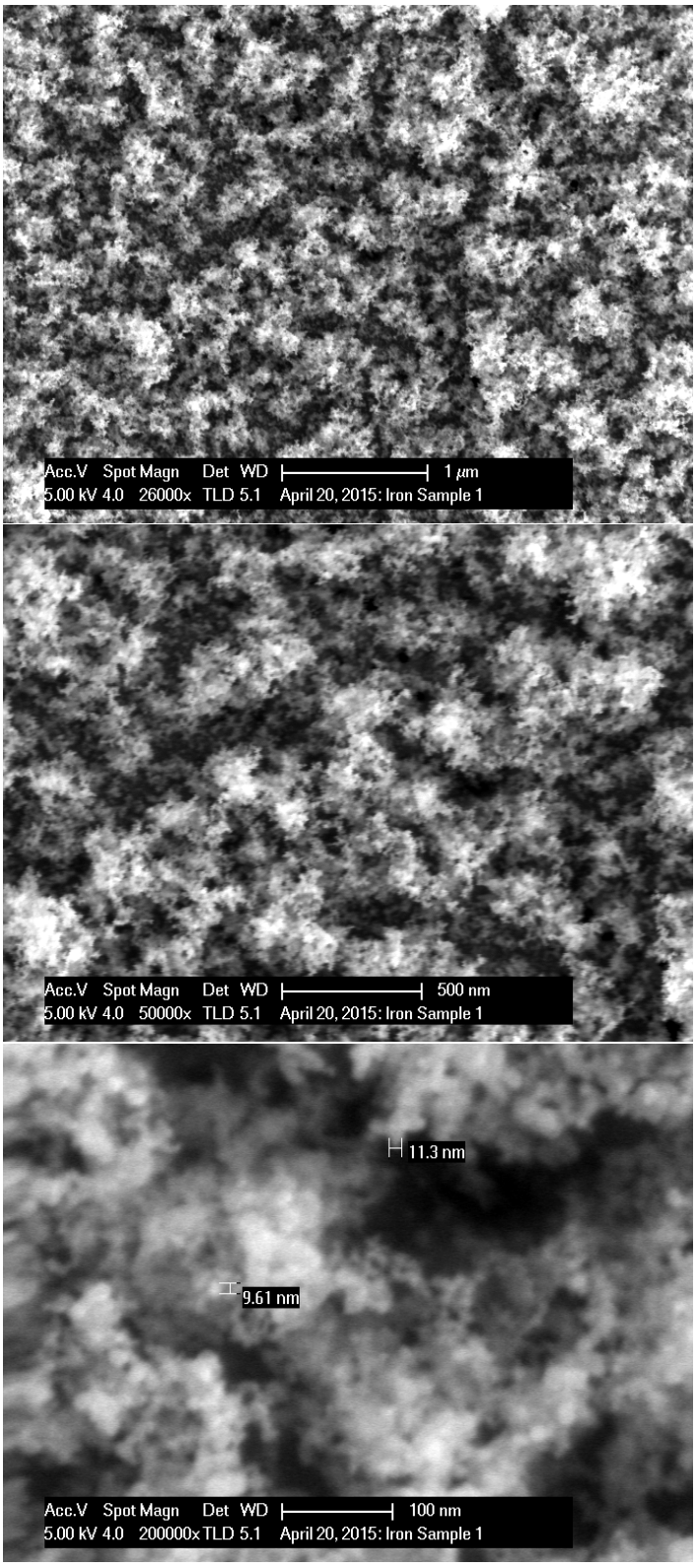

FIGURE B.8: SEM images of an iron DLA film from a top-down view, taken on April 20, 2015

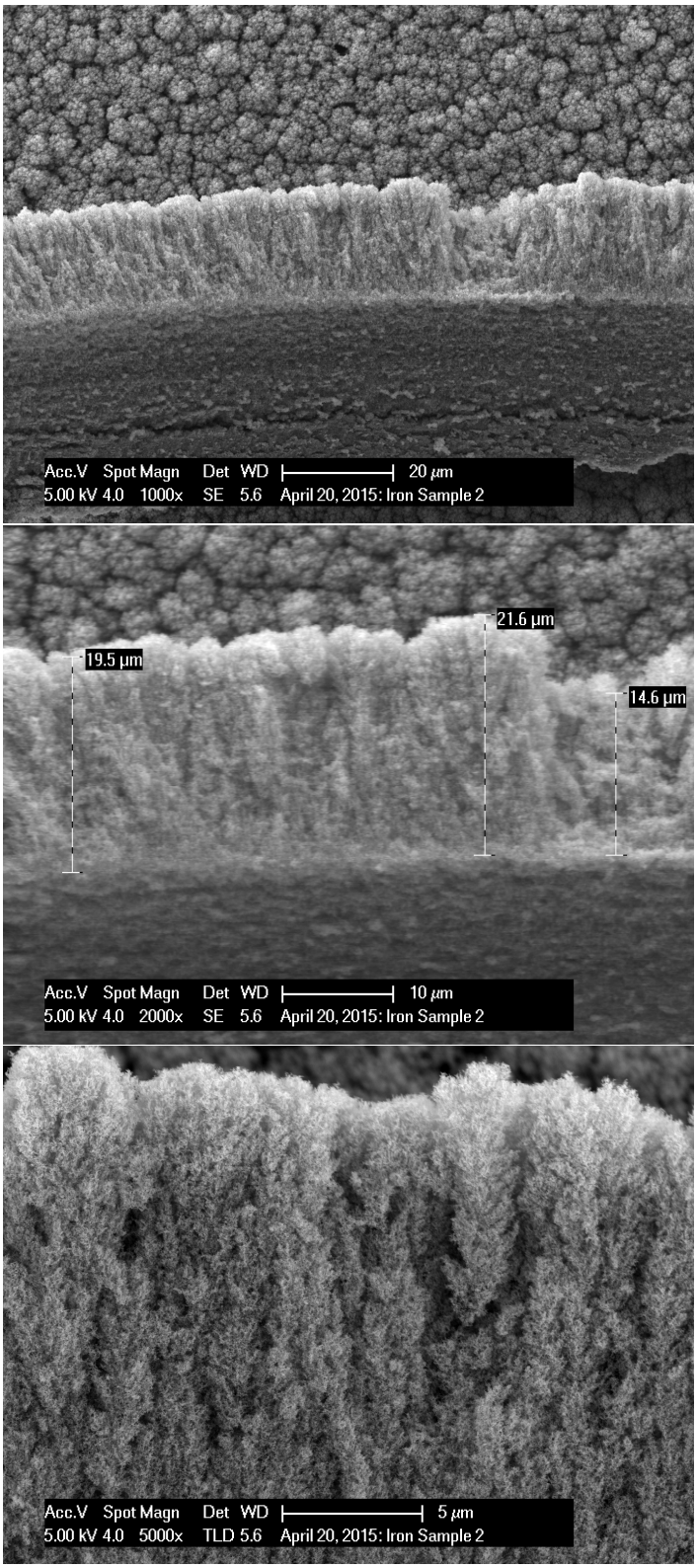

FIGURE B.9: SEM images of an iron DLA film taken on April 20, 2015. More images of this film can be found in the next two figures

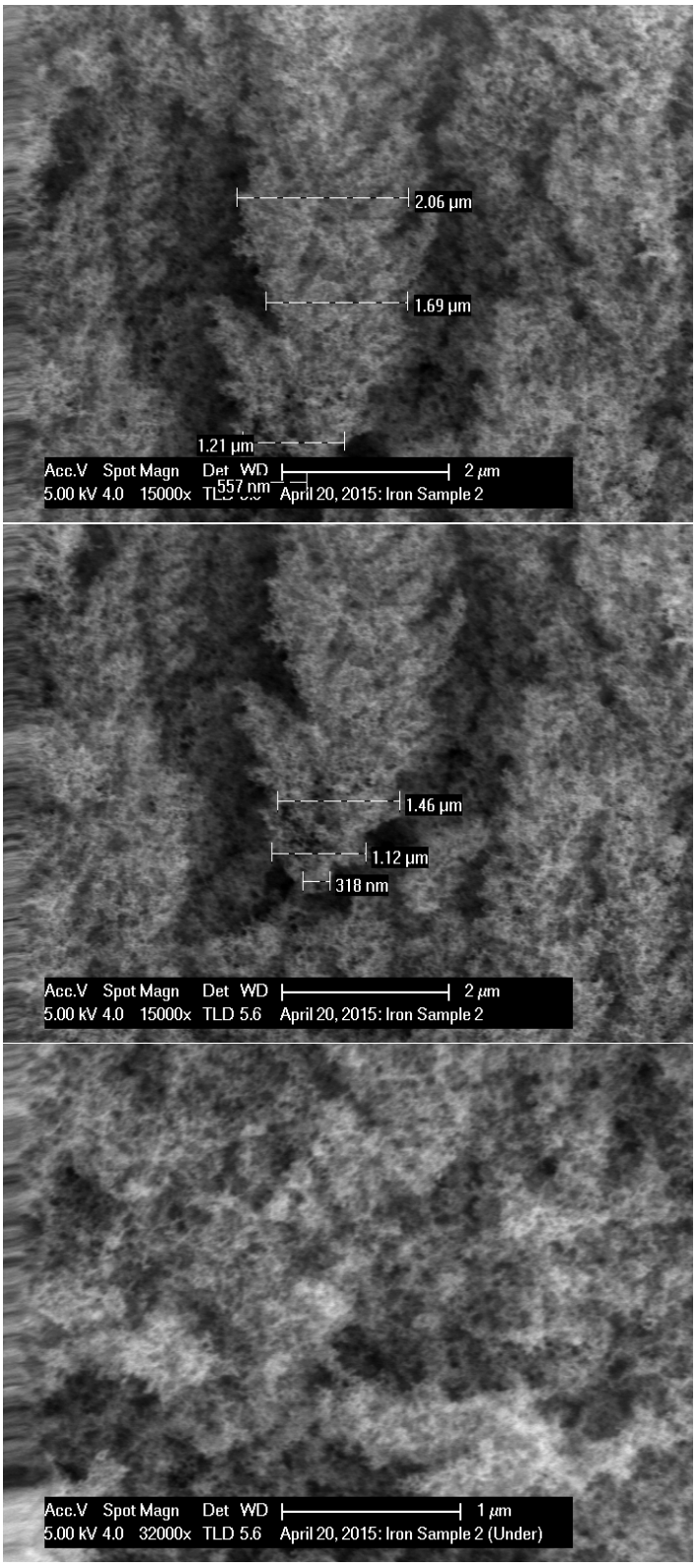

FIGURE B.10: SEM images of an iron DLA film taken on April 20, 2015. More images of this film can be found in the preceding figure and the next figure.

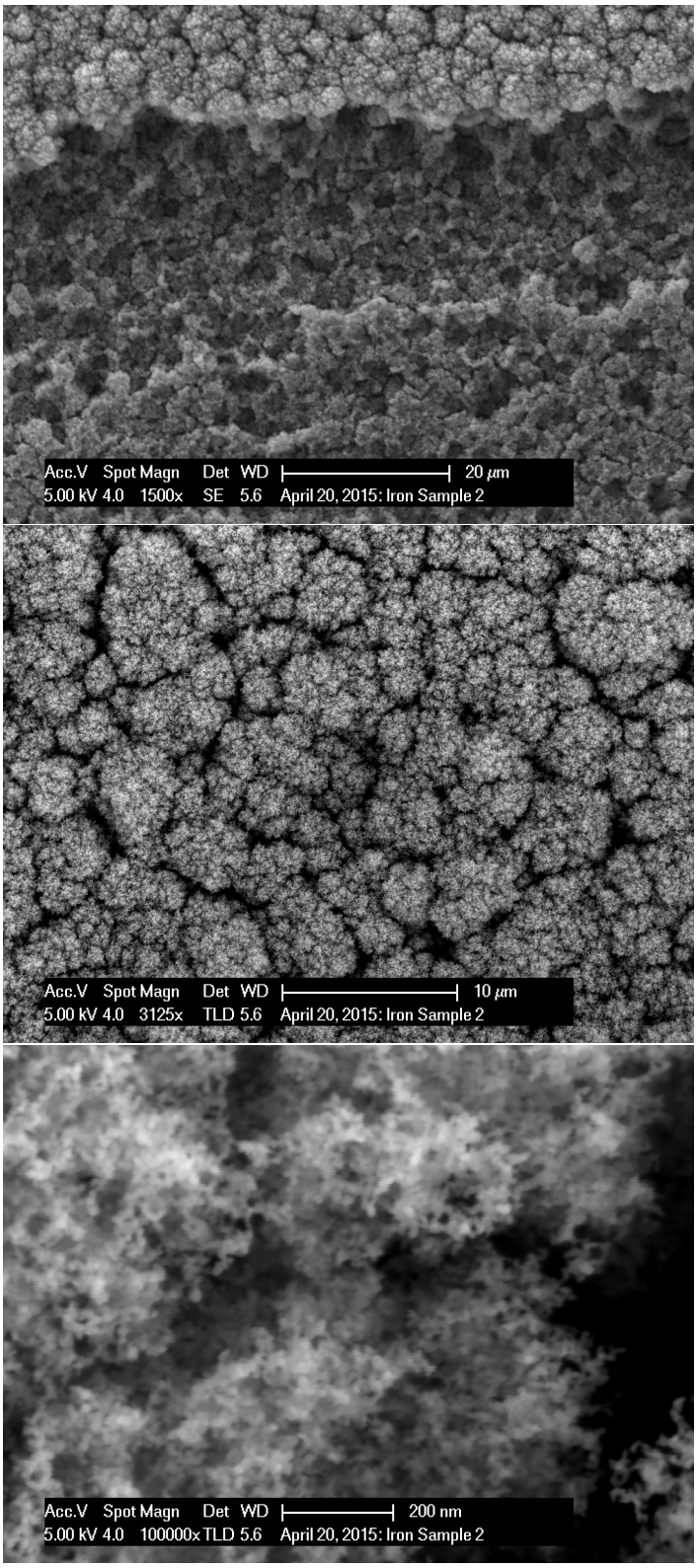

FIGURE B.11: SEM images of an iron DLA film from various views, taken on April 20, 2015. More images of this film can be found in the preceding two figures.

B.3 Silicon Monoxide

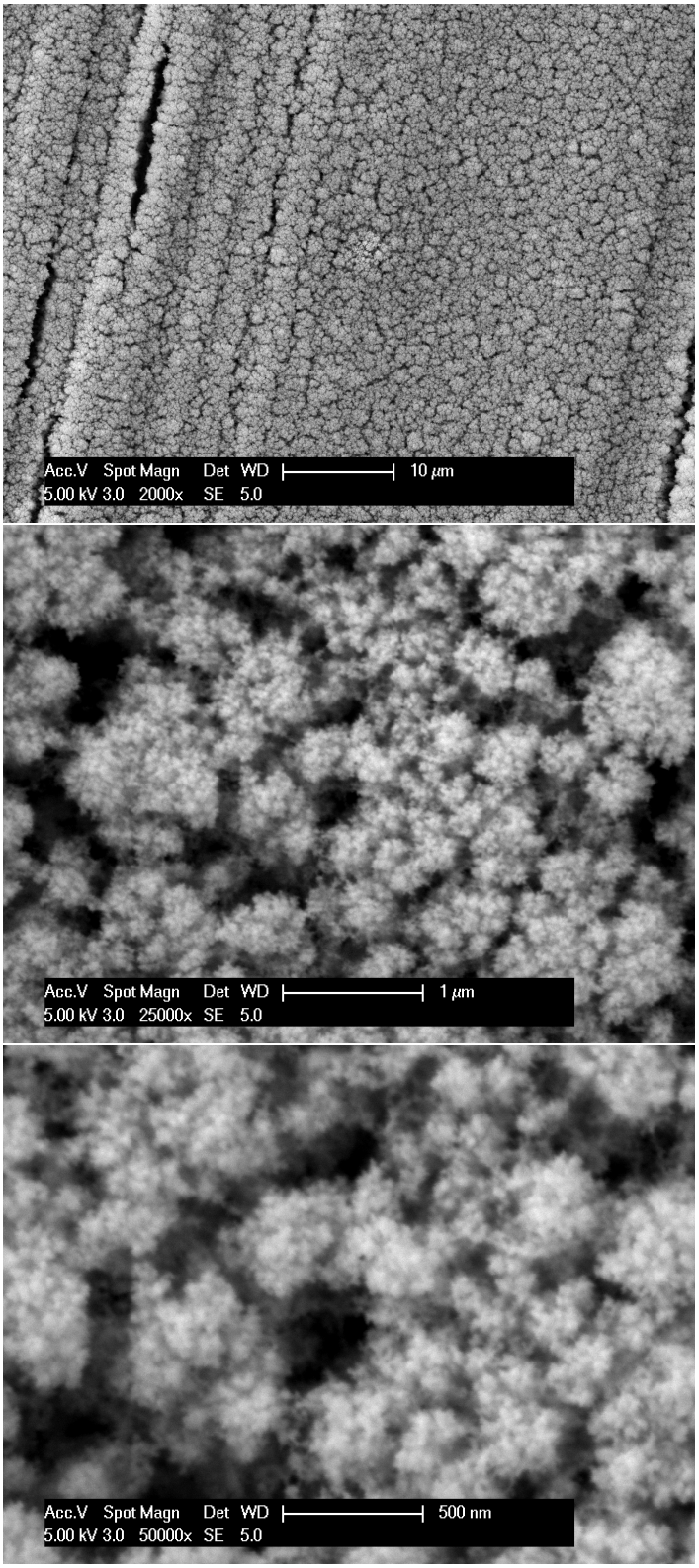

FIGURE B.12: SEM images of silicon monoxide from a top-down view, taken on May 29, 2015.

B.4 Nickel

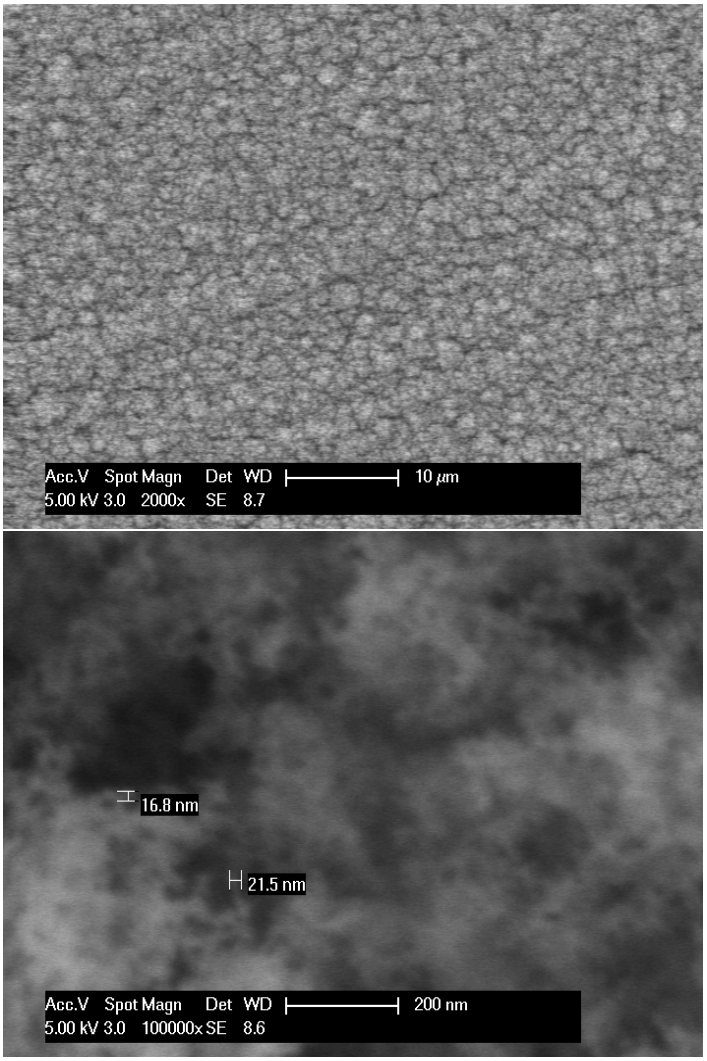

FIGURE B.13: SEM images of Nickel from a top-down view, taken on May 29, 2015.

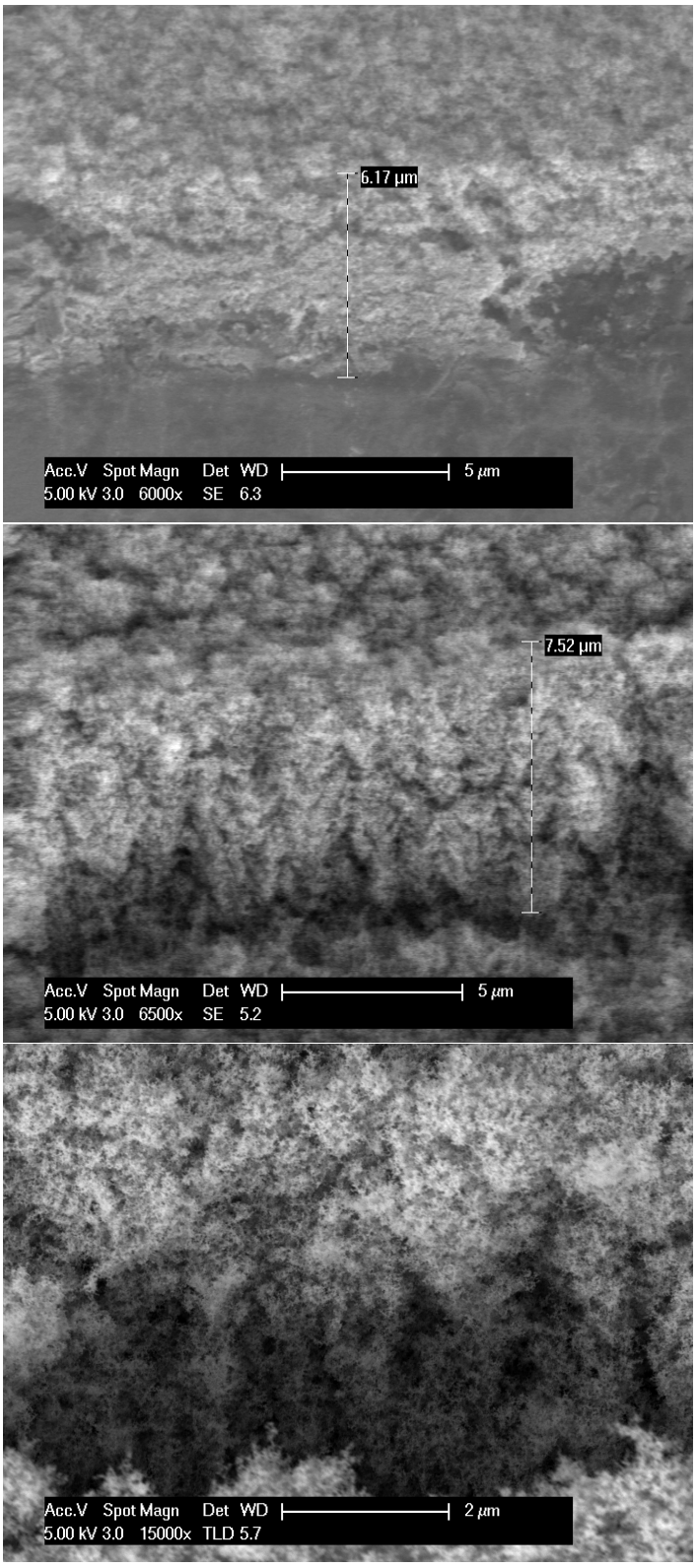

FIGURE B.14: SEM images of Nickel from a side view, taken on May 29, 2015.

B.5 Germanium

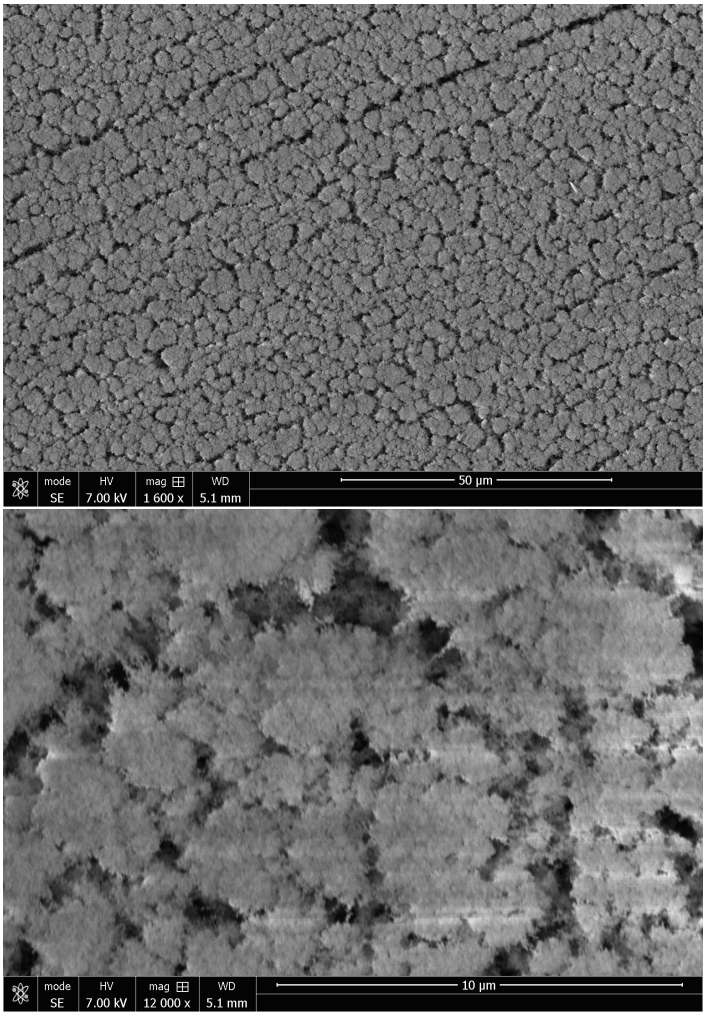

FIGURE B.15: SEM images of a damaged film of germanium. The germanium sample in question had not beed used for photoacoustic data, but had been exposed to a camera flash. Taken on September 12, 2016. More images of this film can be found in the next figure.

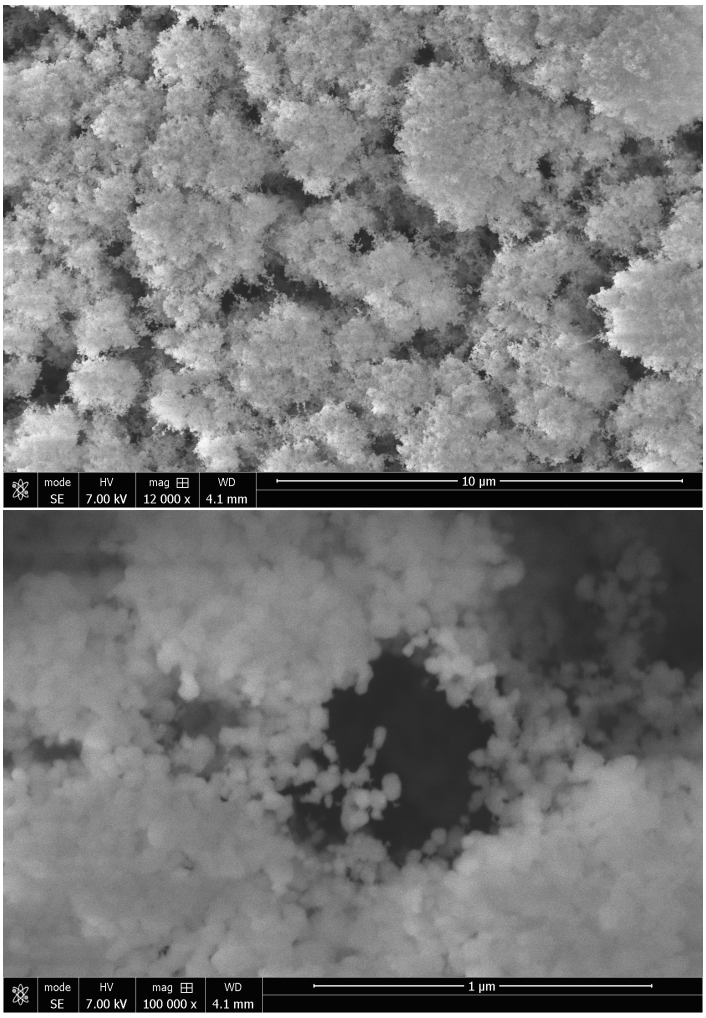

FIGURE B.16: SEM images of a damaged film of germanium. The germanium sample in question had not beed used for photoacoustic data, but had been exposed to a camera flash. Taken on September 12, 2016. More images of this film can be found in the previous figure.

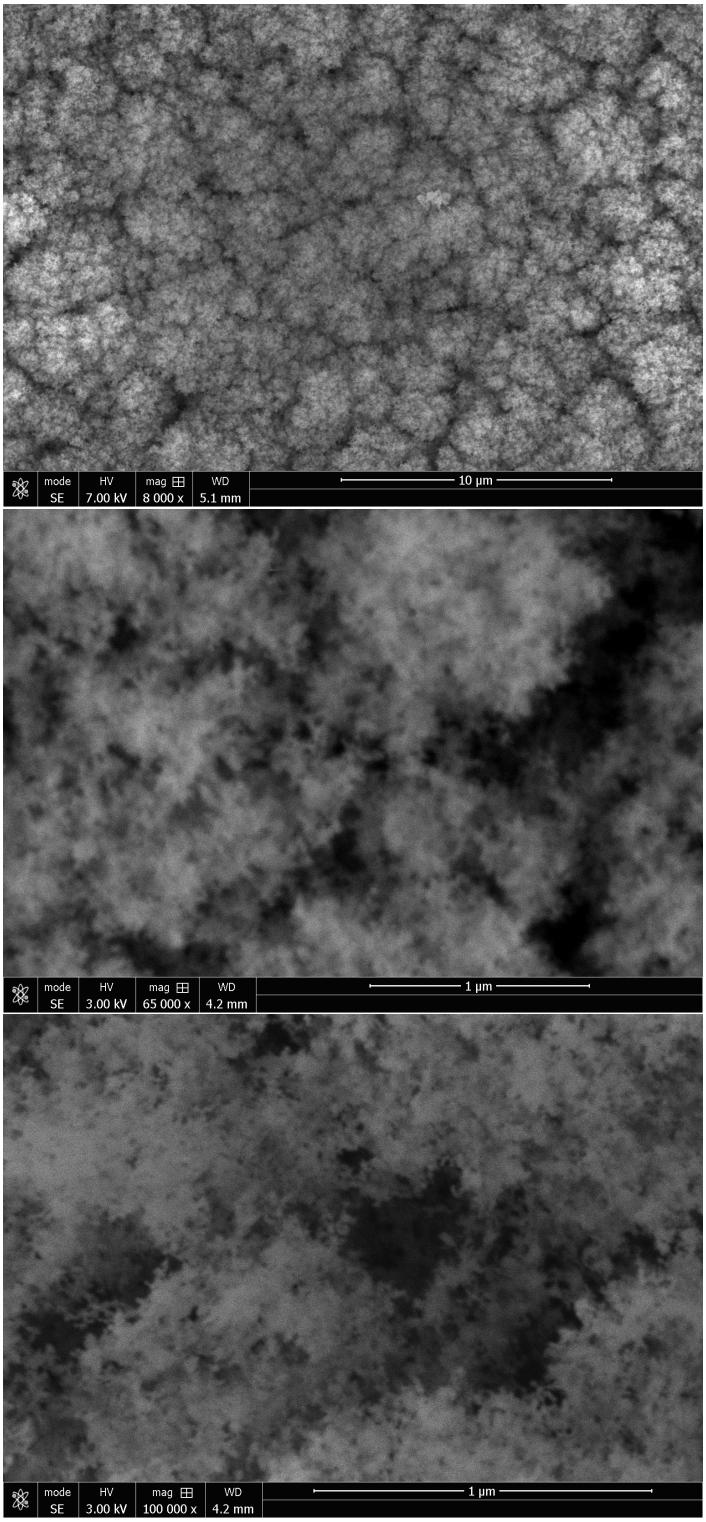

FIGURE B.17: SEM images of a good Germanium sample, taken September 12, 2016

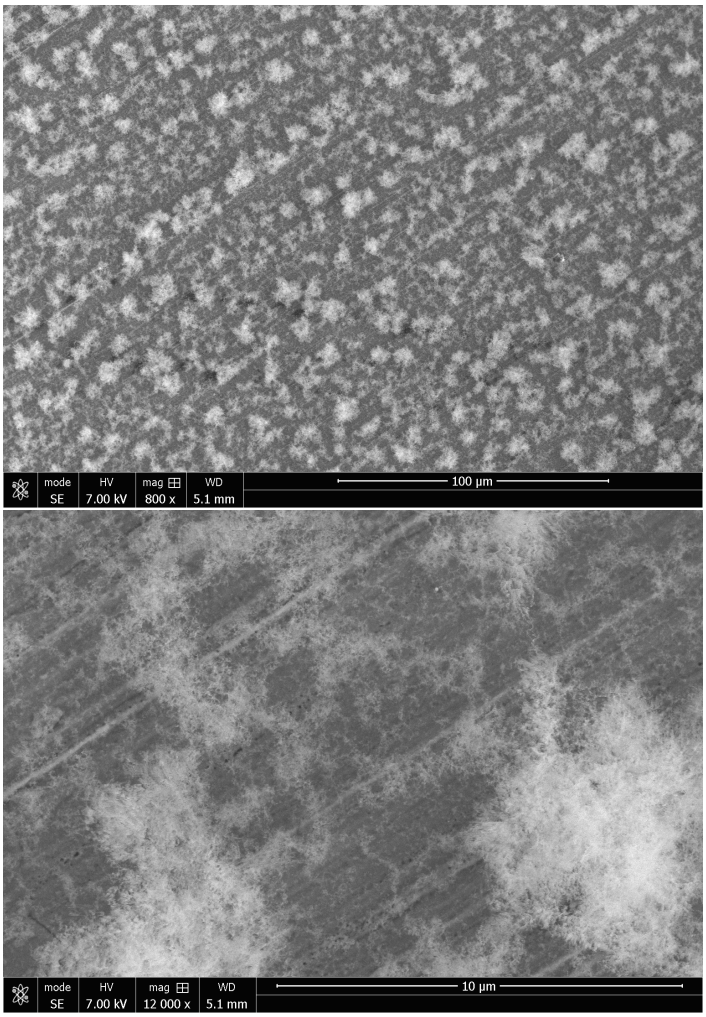

FIGURE B.18: SEMs of a thin germanium DLA film deposited on copper. Taken on September 12, 2016. More images of this film can be found in the next figure.

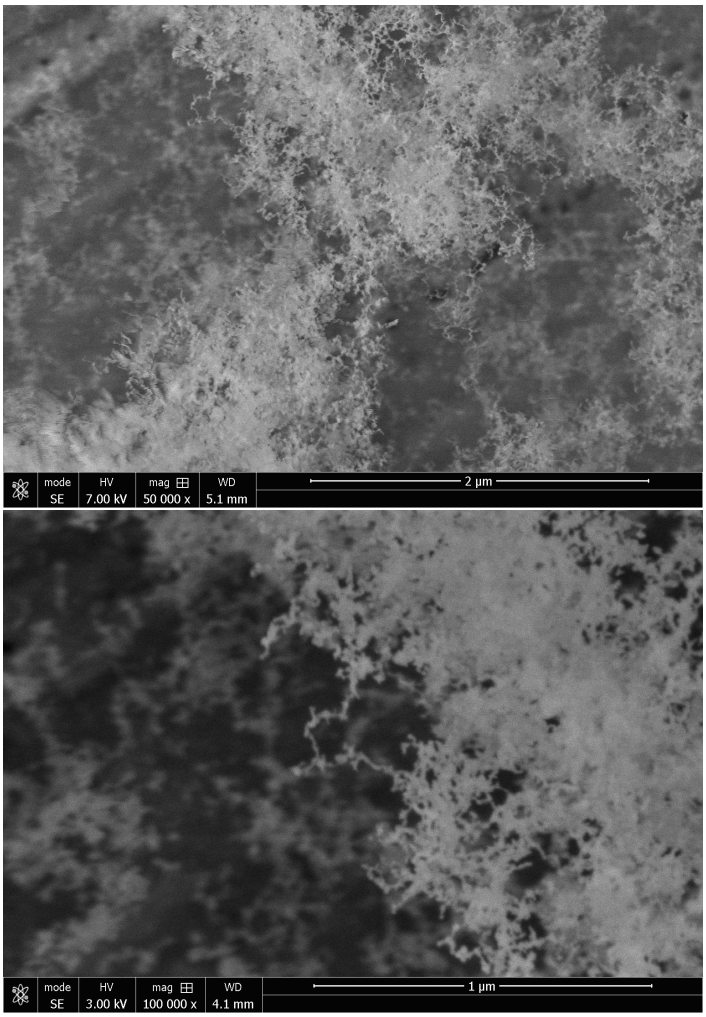

FIGURE B.19: SEMs of a thin germanium DLA film deposited on copper. Taken on September 12, 2016. More images of this film can be found in the previous figure.

B.6 Copper

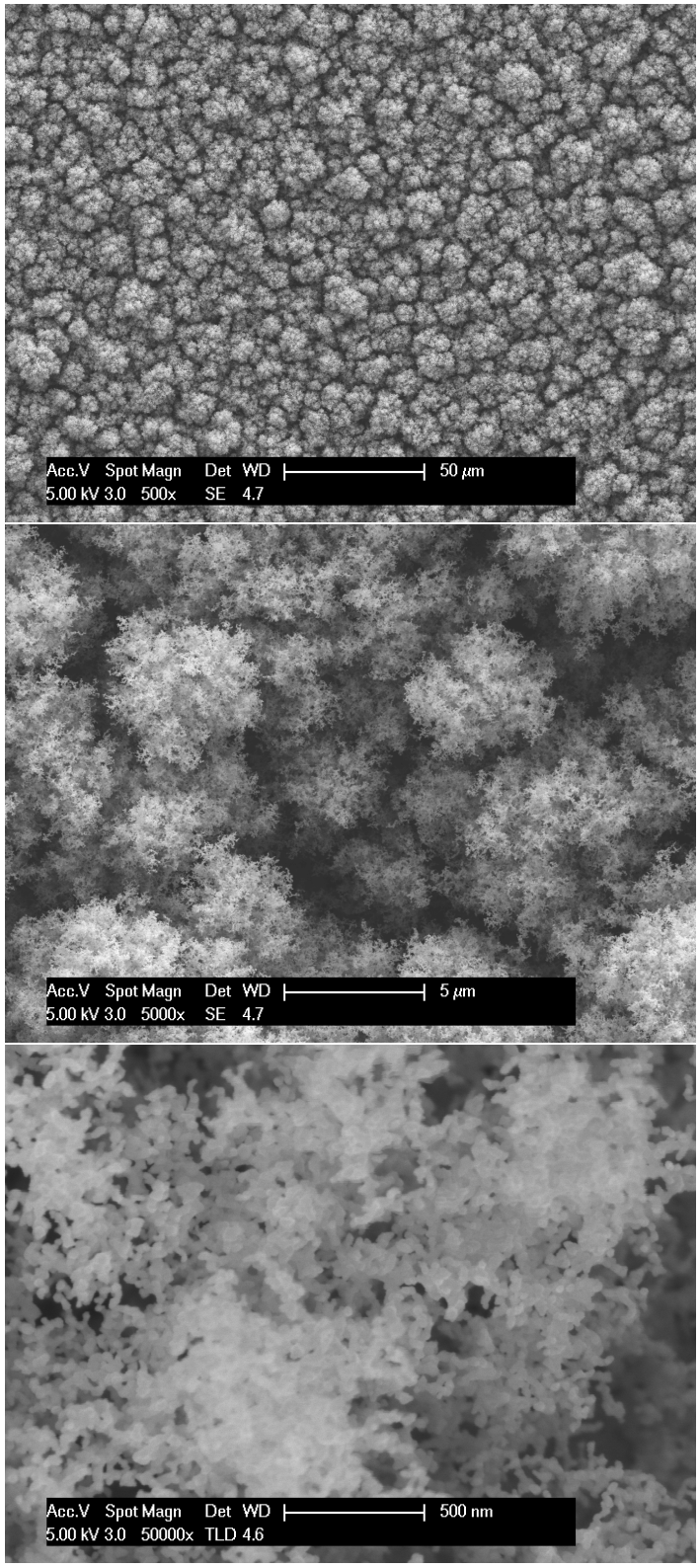

FIGURE B.20: SEM images of a copper DLA film taken by Shura Kotlerman on July 26, 2012. More images of this film can be found in the next figure.

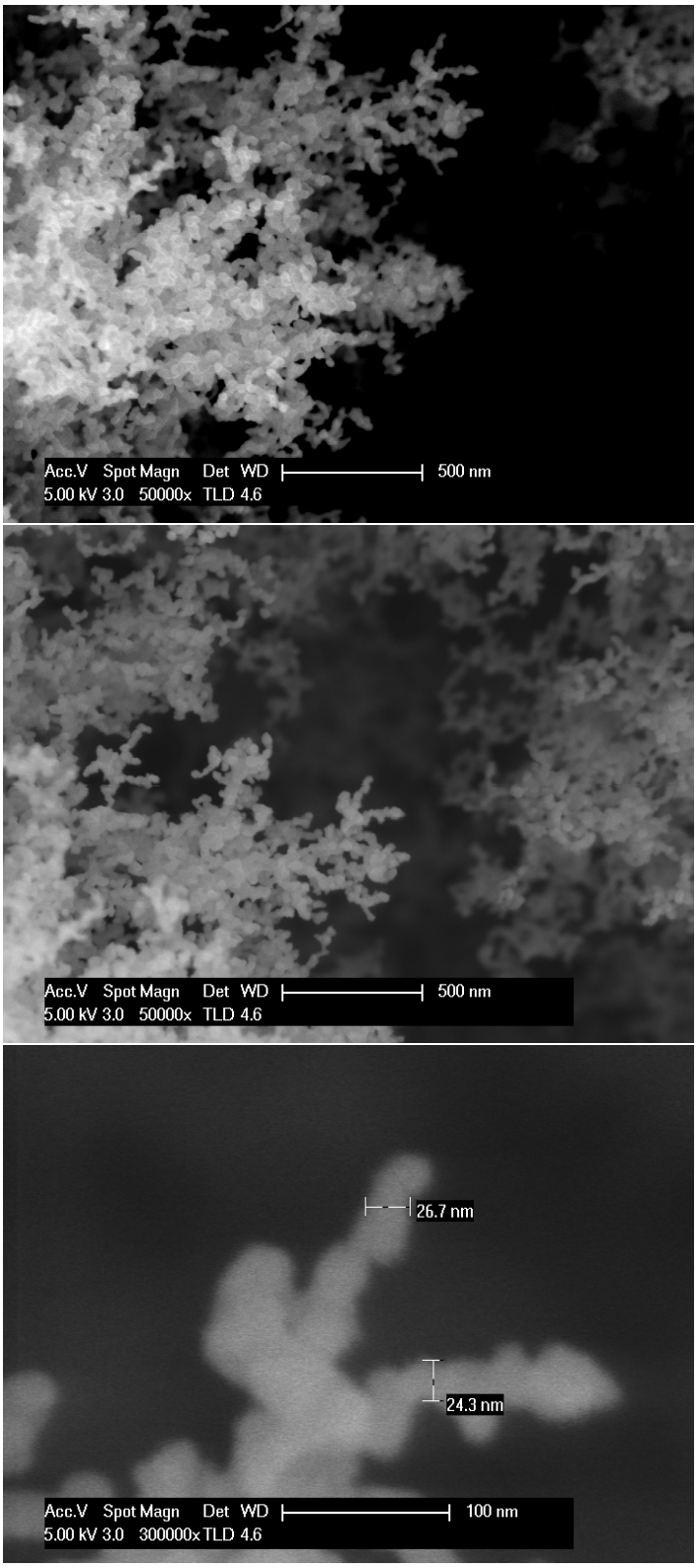

FIGURE B.21: SEM images of a copper DLA film taken by Shura Kotlerman on July 26, 2012. More images of this film can be found in the previous figure.

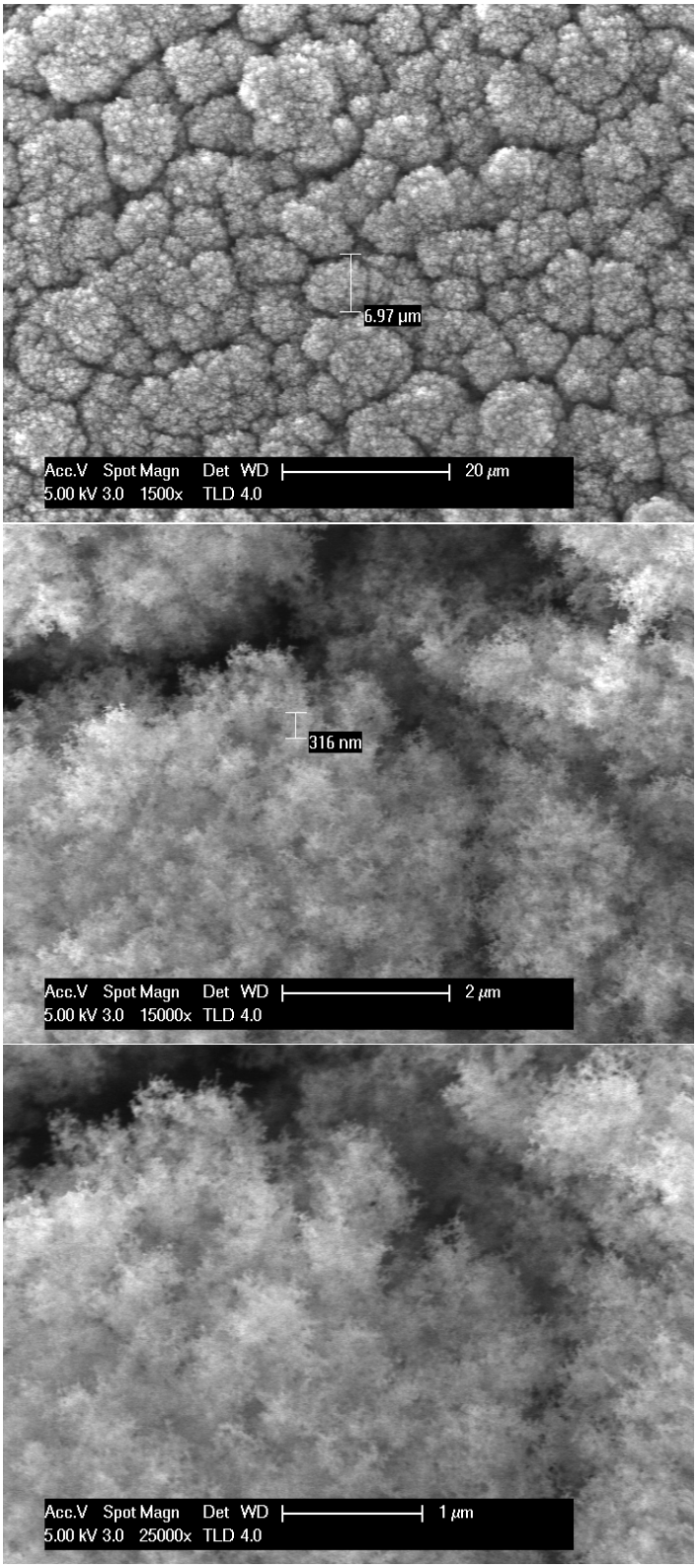

FIGURE B.22: SEM images of a copper DLA film taken by Shura Kotlerman on July 26, 2012. It should be noted that other images of copper taken on the same day (which are shown in the previous two figures) were labeled "RealCu", which could cast some doubt as to whether this film is in fact a copper film. More images of this film can be found in the next figure.

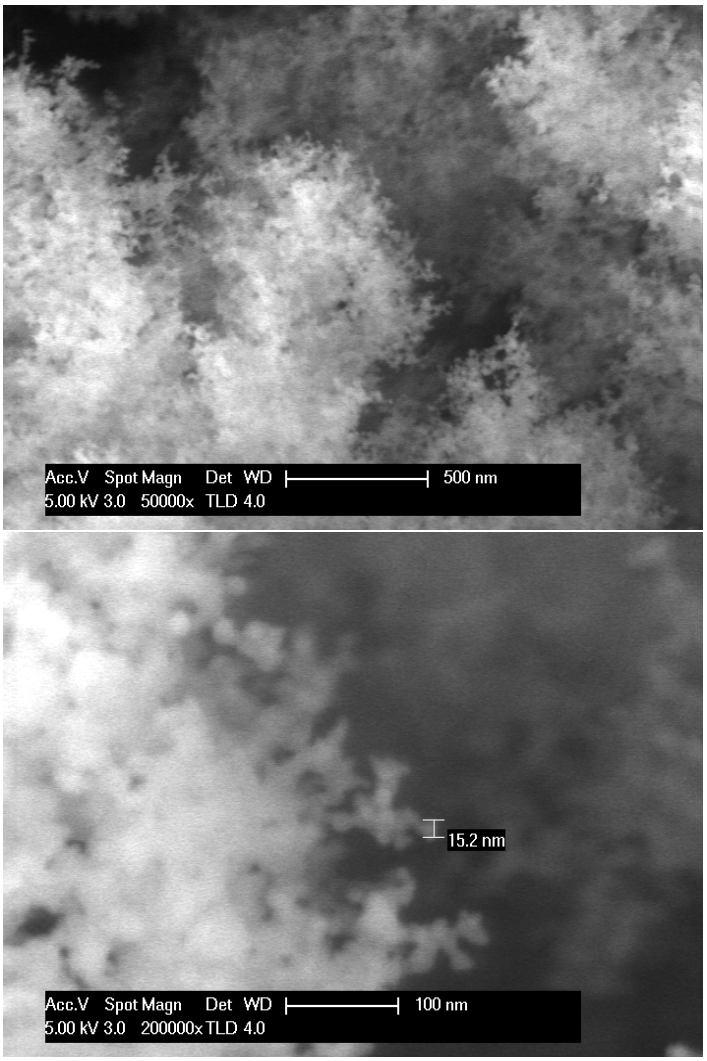

FIGURE B.23: SEM images of a copper DLA film taken by Shura Kotlerman on July 26, 2012. It should be noted that other images of copper taken on the same day were labeled "RealCu", which could cast some doubt as to whether this film is in fact a copper film. More images of this film can be found in the previous figure.

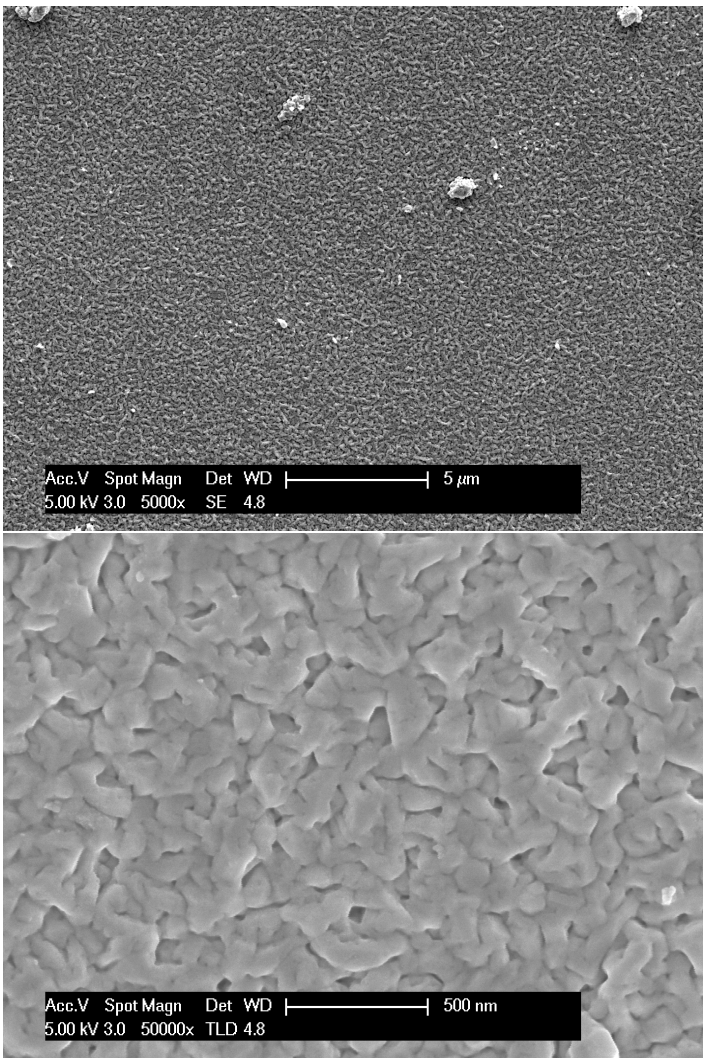

FIGURE B.24: SEM images of a copper sample taken by Shura Kotlerman on July 30, 2012. This sample was deposited on a cover slide in high vacuum, and was formed by three separate depositions.

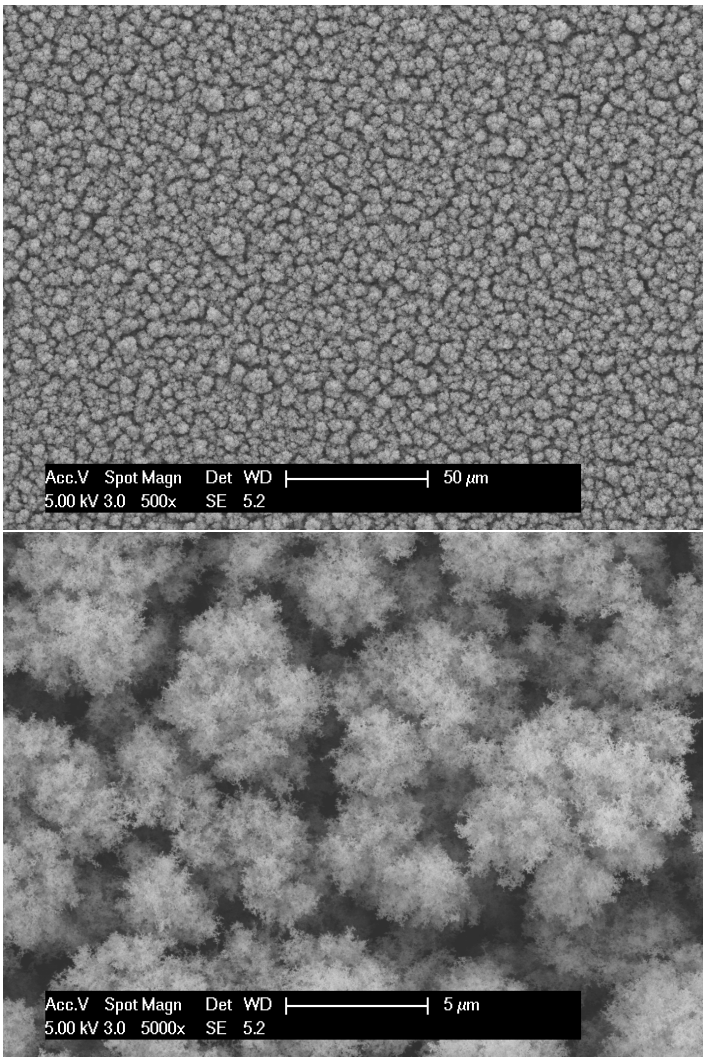

FIGURE B.25: SEM images of a copper sample taken by Shura Kotlerman on July 30, 2012. Sample sample was deposited on a glass slide during a single deposition and was described as a great Cu-black deposition. More images of this sample can be found in the next figure.

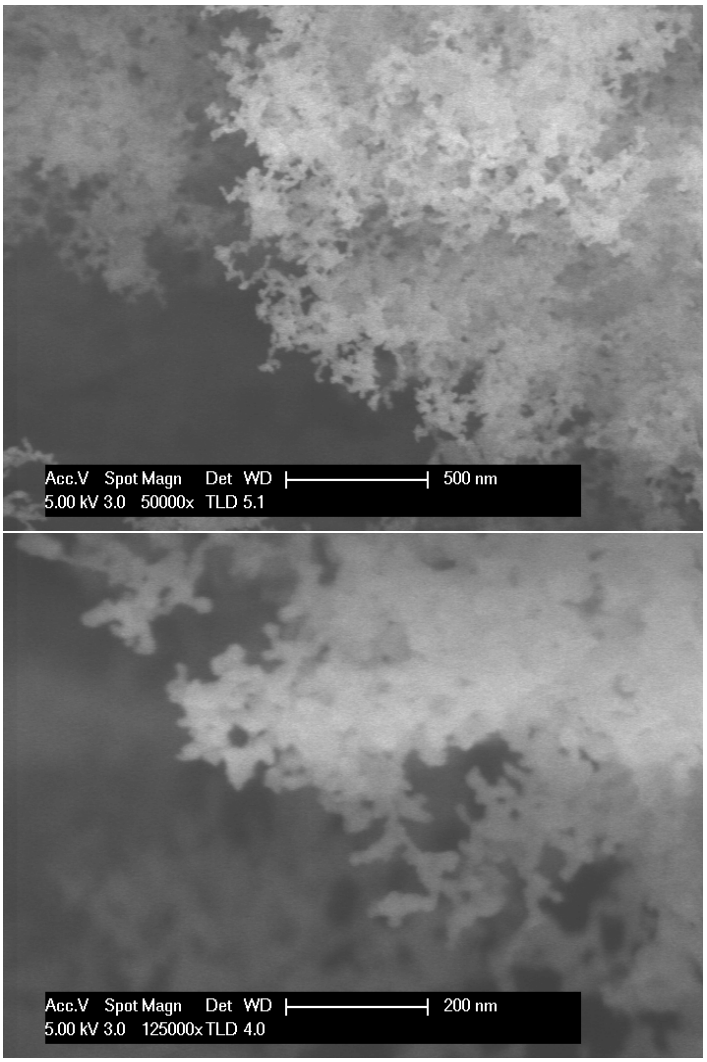

FIGURE B.26: More SEM images of the copper sample shown in the previous figure.

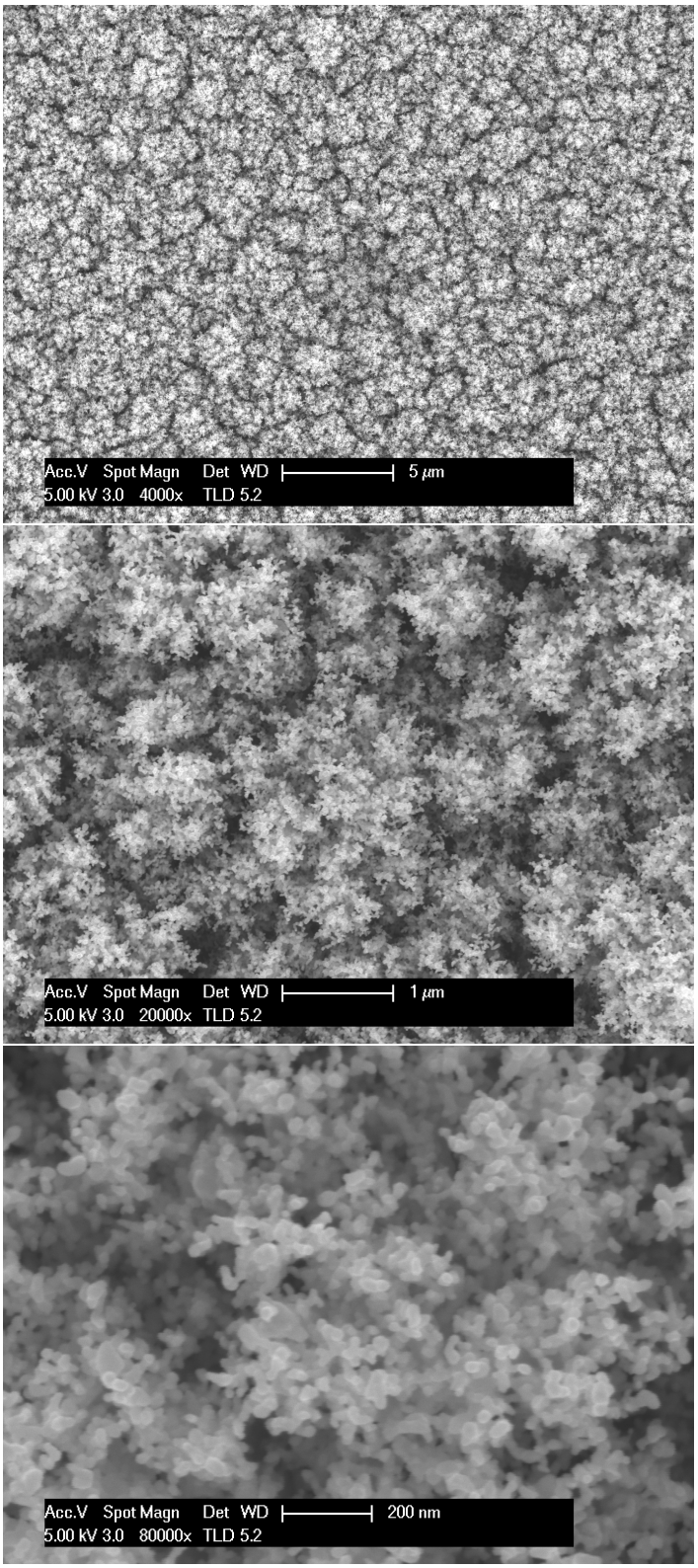

FIGURE B.27: SEM images of a copper DLA film taken by Shura Kotlerman.

B.7 Aluminum and Al_2O_3

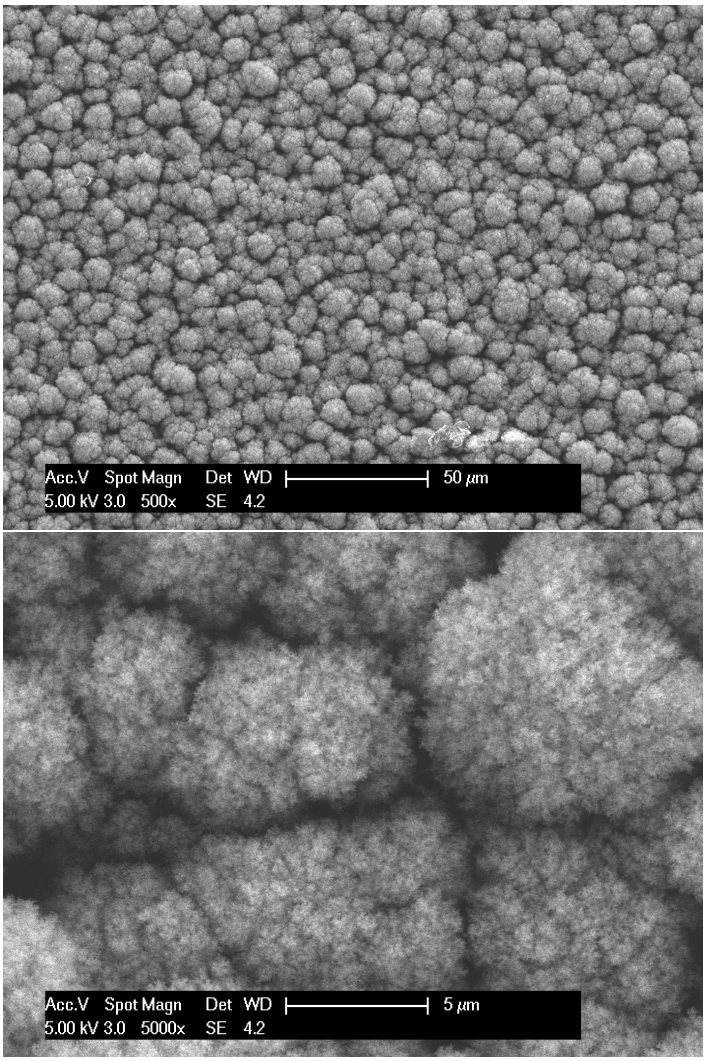

FIGURE B.28: SEM images of an aluminum DLA film taken by Shura Kotlerman on July 26, 2012. More images of this film can be found in the next figure.

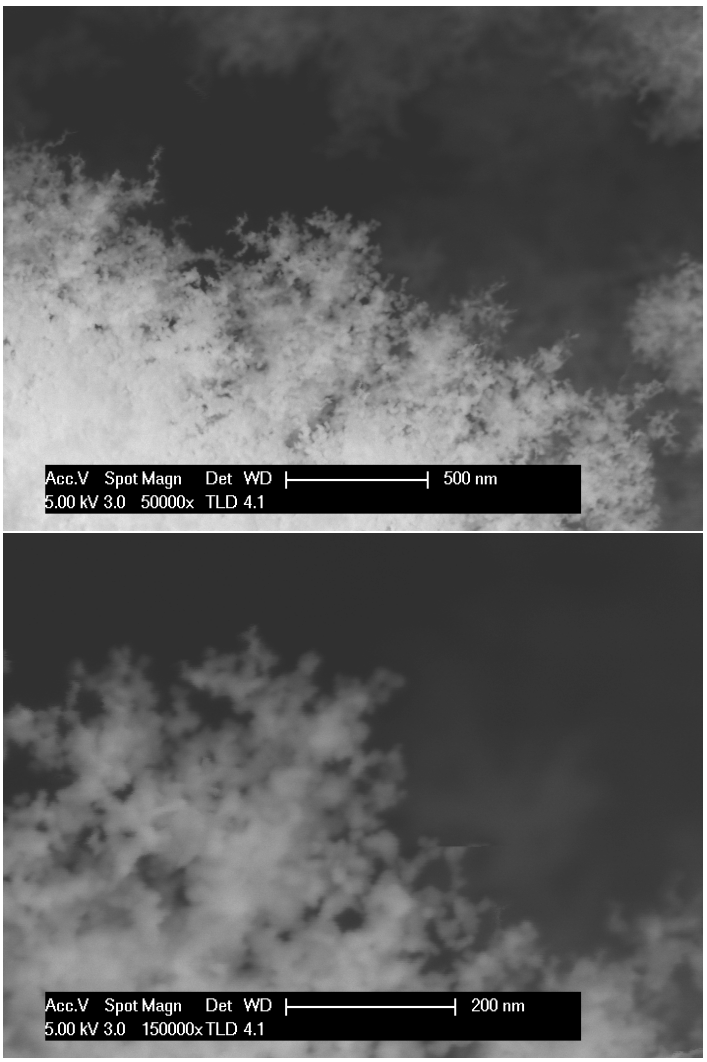

FIGURE B.29: SEM images of an aluminum DLA film taken by Shura Kotlerman on July 26, 2012. More images of this film can be found in the previous figure.

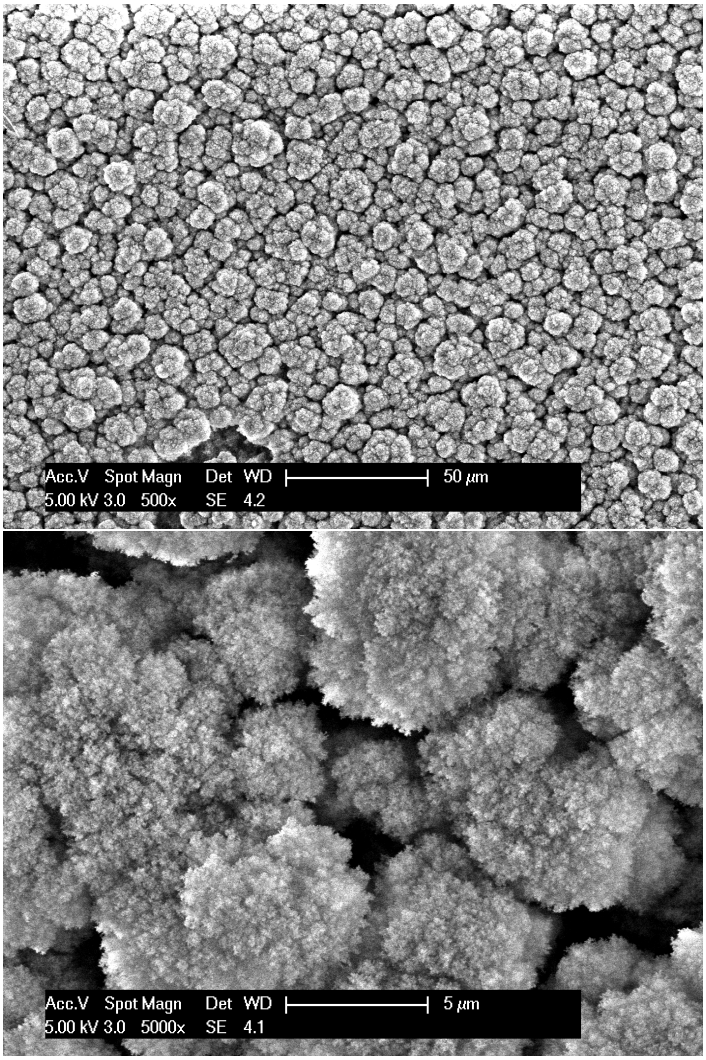

FIGURE B.30: SEM images of a black aluminum DLA film taken by Shura Kotlerman on July 30, 2012.

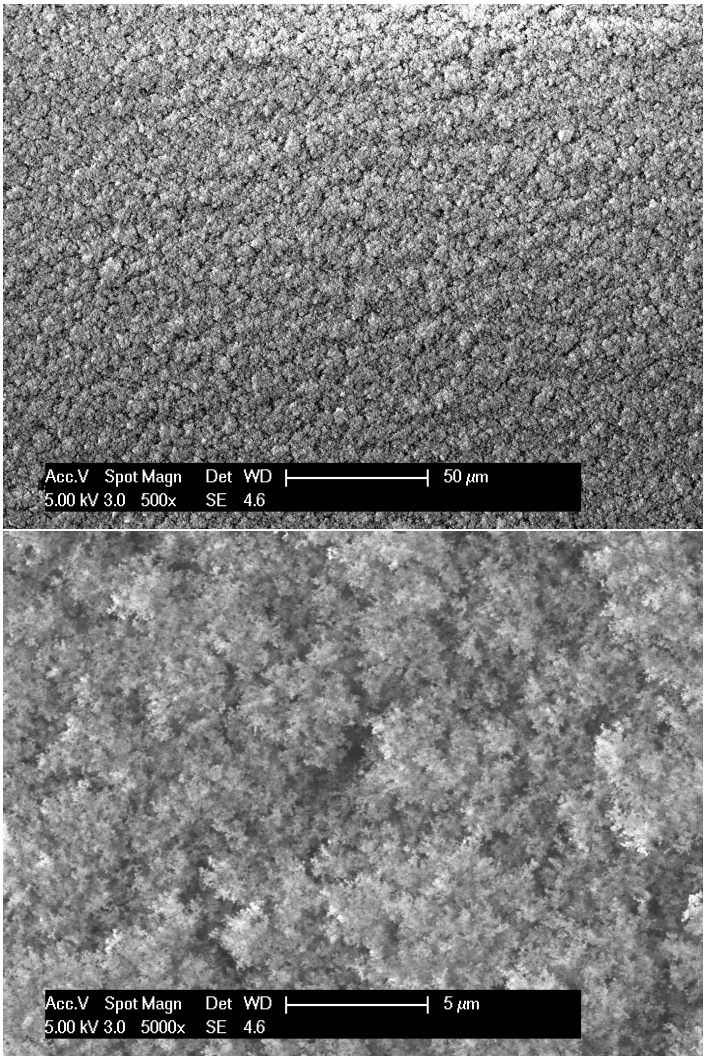

FIGURE B.31: SEM images of a grey aluminum DLA film taken by Shura Kotlerman on July 30, 2012.

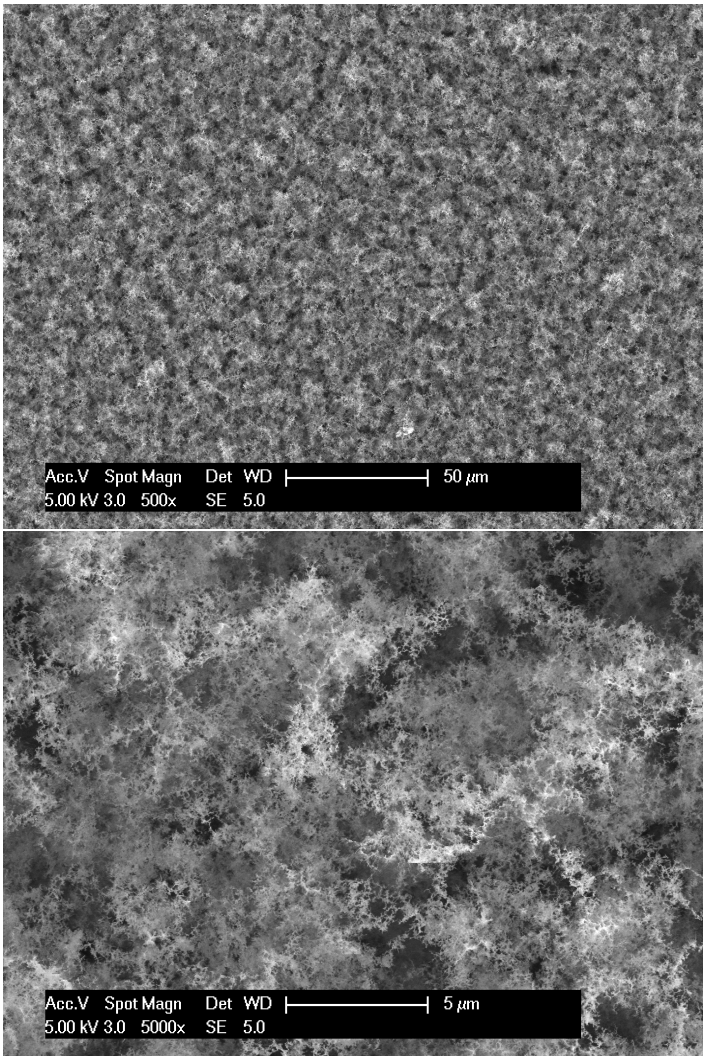

FIGURE B.32: SEM images of $\mathbf{Al_2O_3}$. Images taken by Shura Kotlerman on July 30, 2012.

B.8 Silver

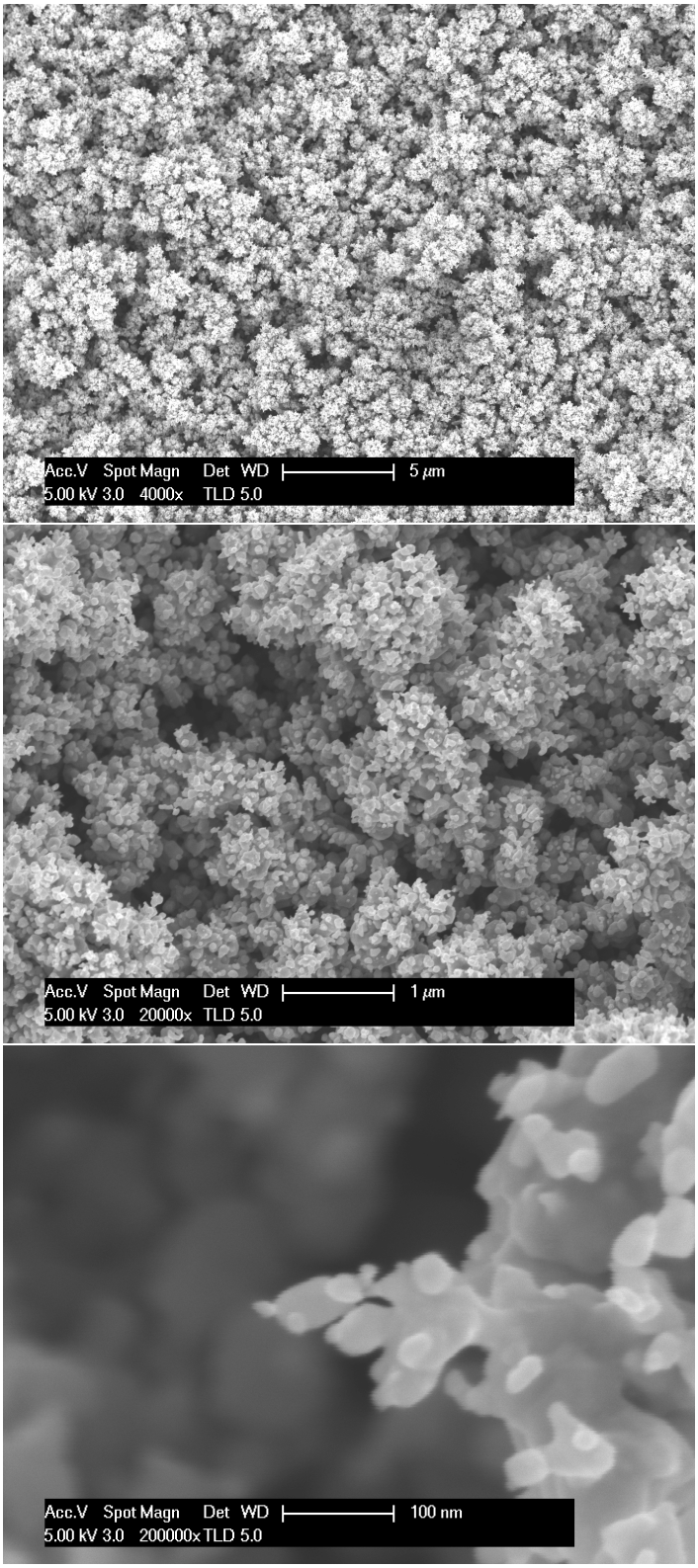

FIGURE B.33: SEM images of a silver DLA film taken by Shura Kotlerman.

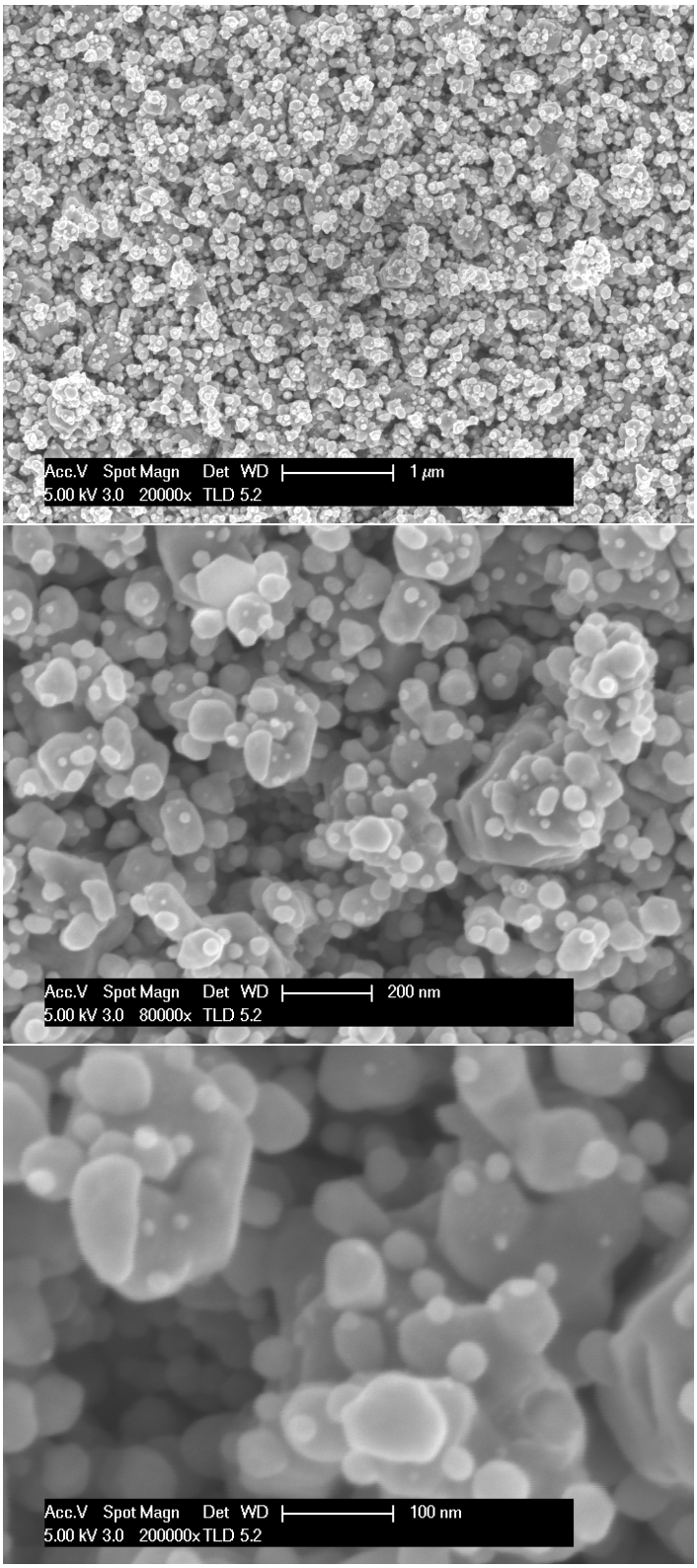

FIGURE B.34: SEM images of a silver DLA film taken by Shura Kotlerman.

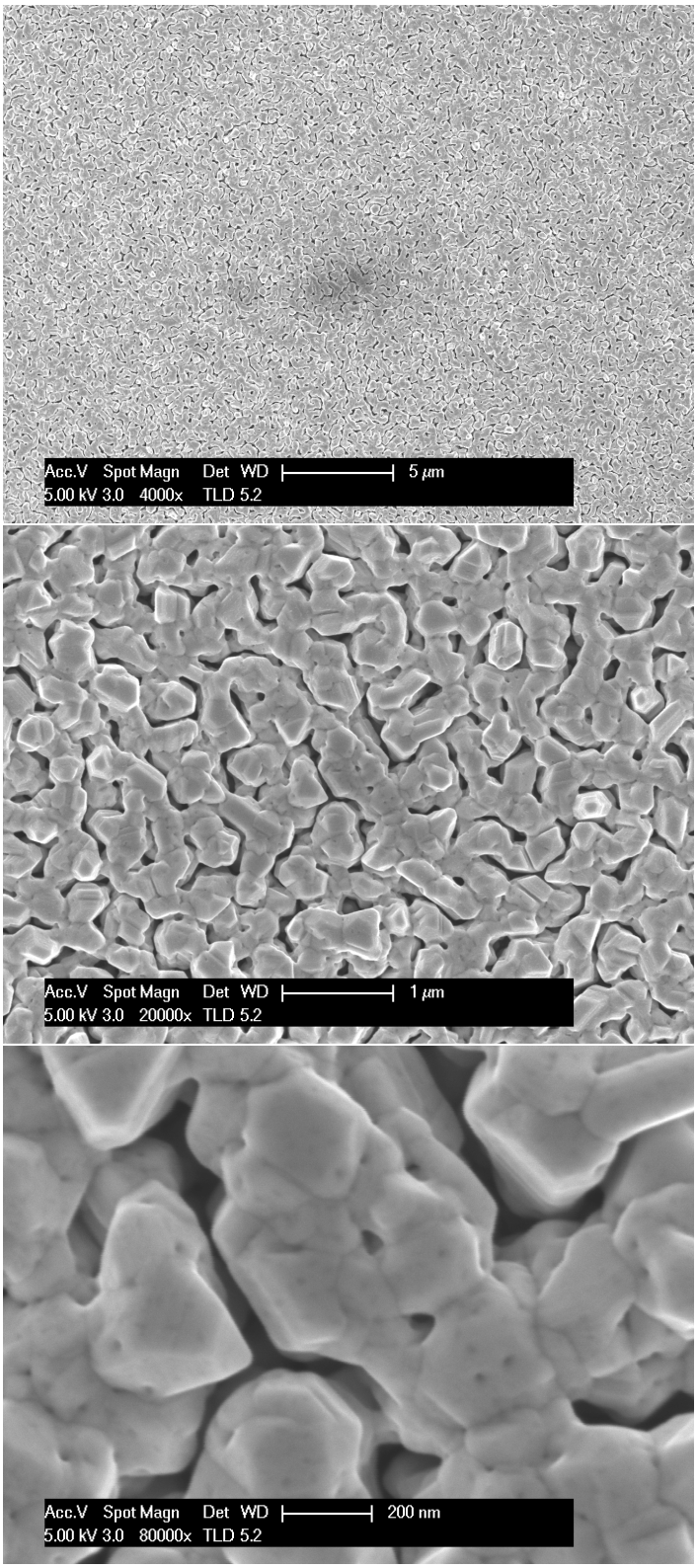

FIGURE B.35: SEM images of a silver DLA film taken by Shura Kotlerman.

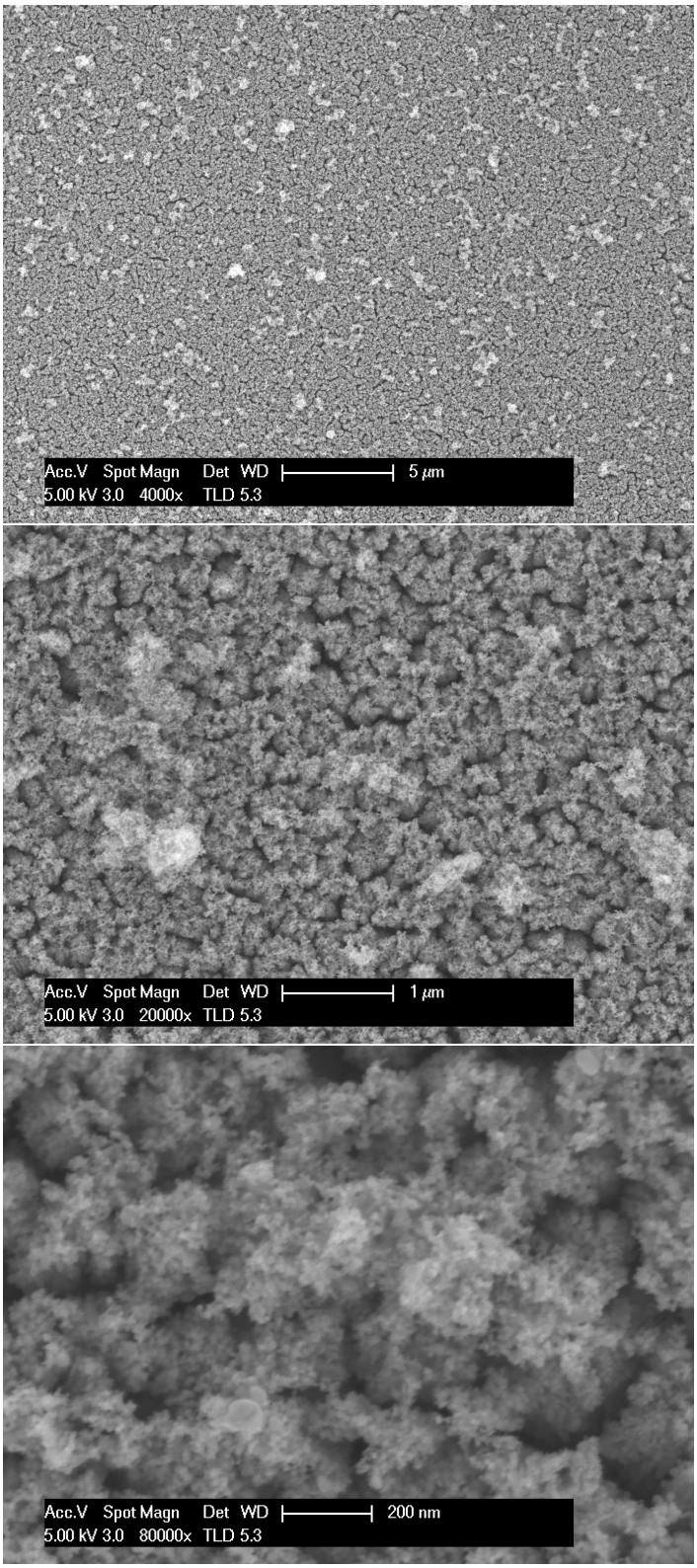

FIGURE B.36: SEM images of a silver DLA film taken by Shura Kotlerman.

B.9 Indium

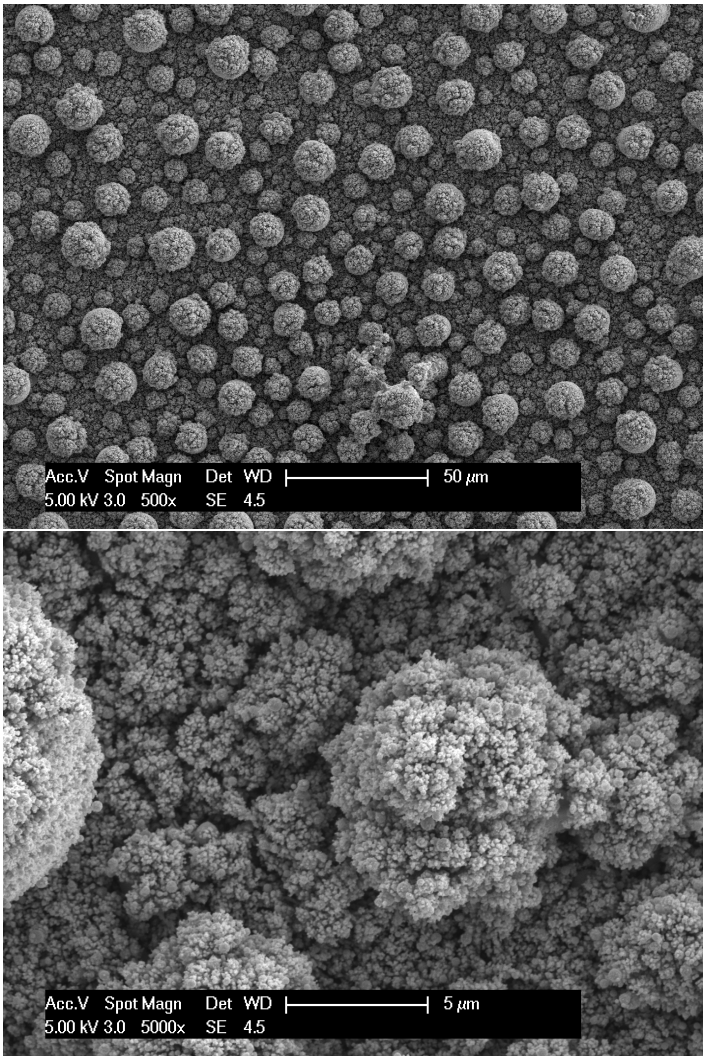

FIGURE B.37: SEM images of an indium DLA film taken by SHura Kotlerman on July 30, 2012. The film was described as black with interference patterns.

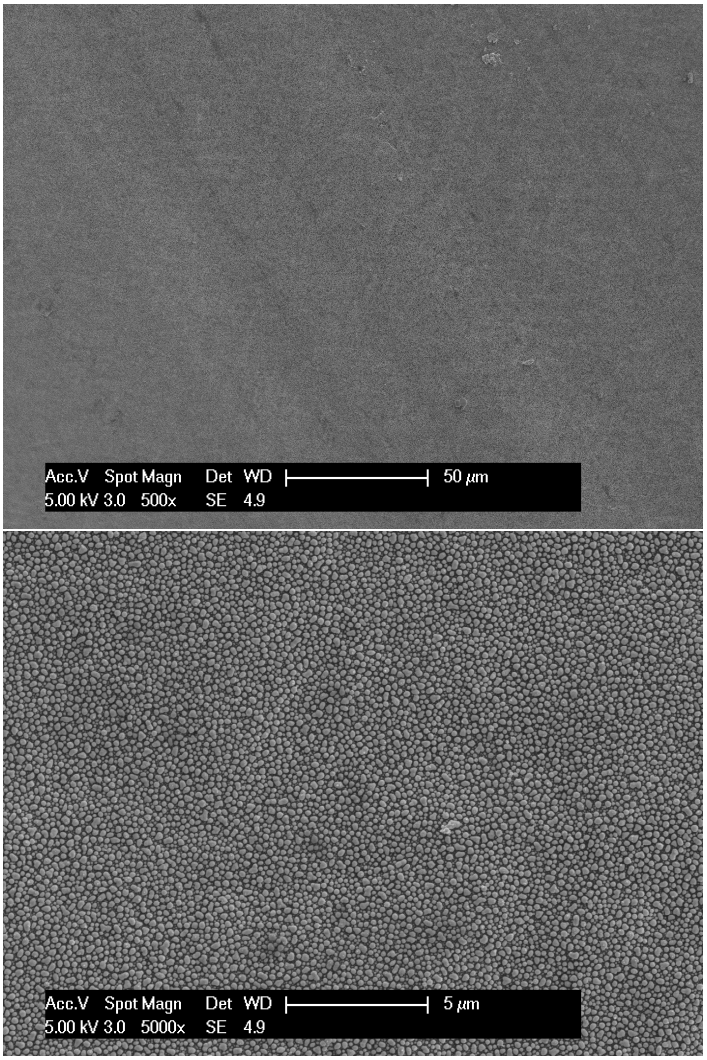

FIGURE B.38: SEM images of an indium DLA film, taken by Shura Kotlerman on July 30, 2012. The film was described as grey and colorful.

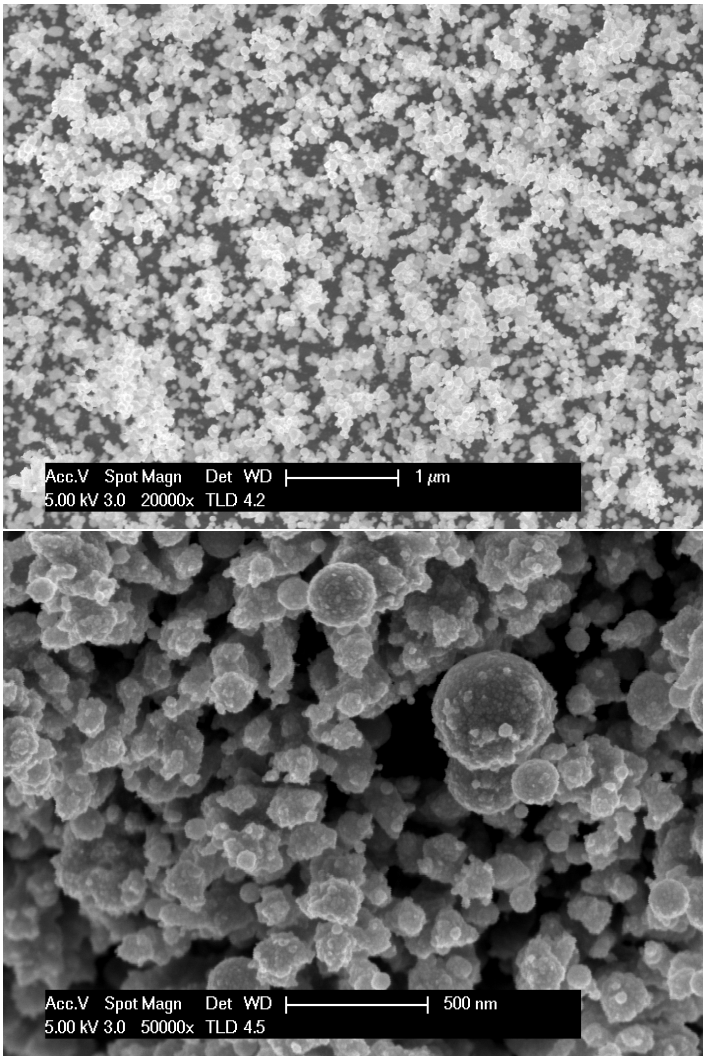

FIGURE B.39: SEM images of two indium DLA films, take by Shura Kotlerman on July 26 and July 30, 2012

B.10 Polyethylene

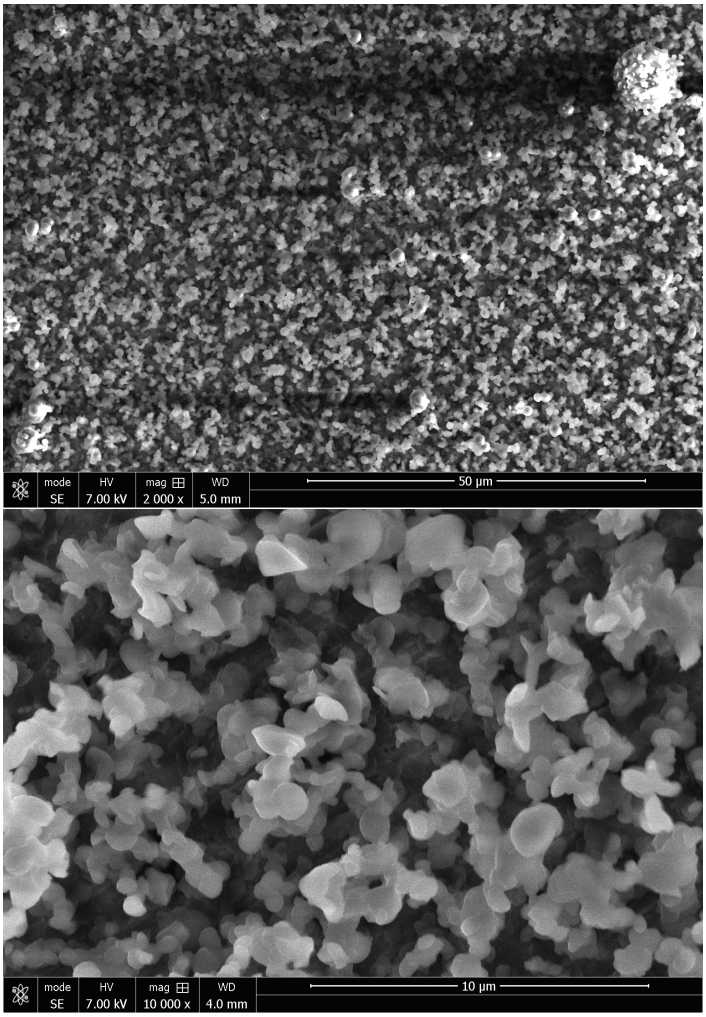

FIGURE B.40: SEMs of a polyethylene sample deposited on copper. Taken on September 12, 2016. More images of this sample can be found in the next figure.

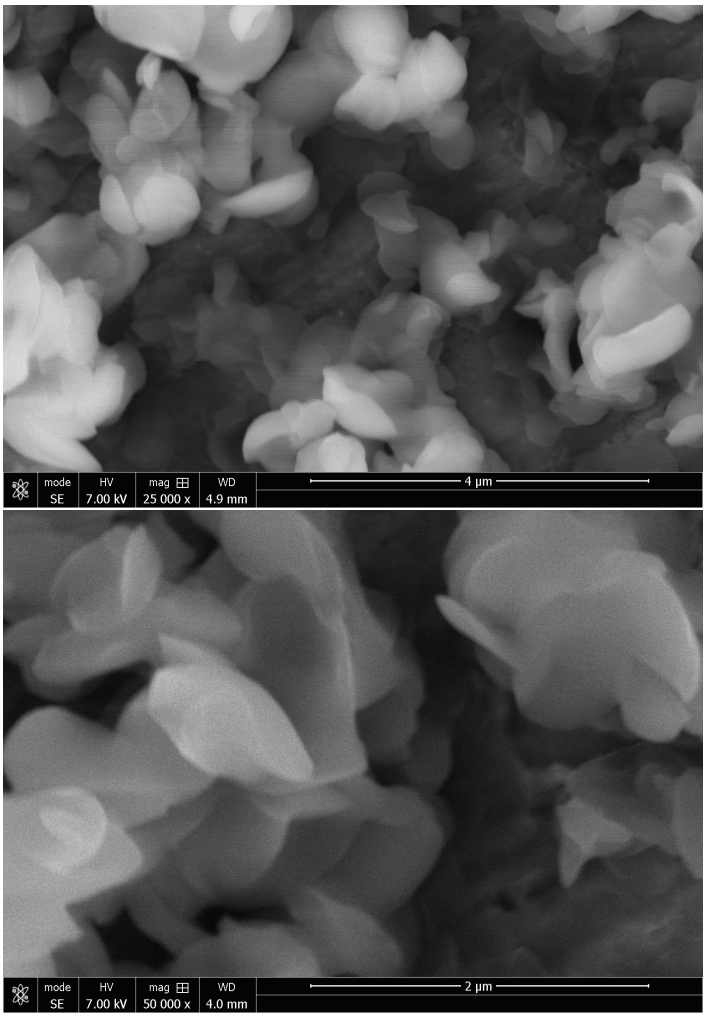

FIGURE B.41: SEMs of a polyethylene sample deposited on copper. Taken on September 12, 2016. More images of this sample can be found in the previous figure.

Appendix C

Programs

This appendix contains the vast majority of the programs used in the course of this research. Instructions on how to use each program or class of program will be provided before the code itself. Note that a red arrow at the beginning of a line indicates that the previous line of code is being continued, and does not actually appear in the programs themselves.

C.1 FPGA Programs

C.1.1 phased_array_control_backup.v

This program is an early example of the program that sets the phase pattern of the FPGA. This file is now used as a reference program used to construct the correct verilog file for a given sound pattern. Direct interaction with this code is not recommended since the contents of this file are used as a reference in other programs. The version that is actually uploaded to the FPGA is phased_array_control.v, and although one can alter some parameters directly in this program, it is still recommended that one use the other programs to generate appropriate versions of this program.

```
// --------------------------------------------------------------------
// Copyright (c) 2009 by Terasic Technologies Inc.
// --------------------------------------------------------------------
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
// --------------------------------------------------------------------
//
//                     Terasic Technologies Inc
//                     356 Fu-Shin E. Rd Sec. 1. JhuBei City,
//                     HsinChu County, Taiwan
//                     302
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
//
// --------------------------------------------------------------------
//
// Major Functions:     DE0 Default
//
// --------------------------------------------------------------------
//
// Revision History :
// --------------------------------------------------------------------
//   Ver  :| Author            :| Mod. Date :| Changes Made:


// --------------------------------------------------------------------

module phased_array_control
(
```

```
        //////////////////    Clock Input             //////////////////
        CLOCK_50,                                     //      50 MHz
        CLOCK_50_2,                                   //      50 MHz
        //////////////////    Push Button             //////////////////
        BUTTON,                                       //      Pushbutton[2:0]
        //////////////////    DPDT Switch             //////////////////
        SW,                                           //      Toggle Switch[9:0]
        //////////////////    7-SEG Dispaly   //////////////////
        HEX0_D,                                       //      Seven Segment Digit 0
        HEX0_DP,                                      //      Seven Segment Digit DP 0
        HEX1_D,                                       //      Seven Segment Digit 1
        HEX1_DP,                                      //      Seven Segment Digit DP 1
        HEX2_D,                                       //      Seven Segment Digit 2
        HEX2_DP,                                      //      Seven Segment Digit DP 2
        HEX3_D,                                       //      Seven Segment Digit 3
        HEX3_DP,                                      //      Seven Segment Digit DP 3
        ////////////////////////        LED             ////////////////////////
        LEDG,                                         //      LED Green[9:0]
        ////////////////////////        UART    ////////////////////////
        UART_TXD,                                     //      UART Transmitter
        UART_RXD,                                     //      UART Receiver
        UART_CTS,                                     //      UART Clear To Send
        UART_RTS,                                     //      UART Request To Send
        //////////////////    SDRAM Interface         ////////////////
        DRAM_DQ,                                      //      SDRAM Data bus 16 Bits
        DRAM_ADDR,                                    //      SDRAM Address bus 13 Bits
        DRAM_LDQM,                                    //      SDRAM Low-byte Data Mask
        DRAM_UDQM,                                    //      SDRAM High-byte Data Mask
        DRAM_WE_N,                                    //      SDRAM Write Enable
        DRAM_CAS_N,                                   //      SDRAM Column Address Strobe
        DRAM_RAS_N,                                   //      SDRAM Row Address Strobe
        DRAM_CS_N,                                    //      SDRAM Chip Select
        DRAM_BA_0,                                    //      SDRAM Bank Address 0
        DRAM_BA_1,                                    //      SDRAM Bank Address 1
        DRAM_CLK,                                     //      SDRAM Clock
        DRAM_CKE,                                     //      SDRAM Clock Enable
        //////////////////    Flash Interface         ////////////////
        FL_DQ,                                        //      FLASH Data bus 15 Bits
        FL_DQ15_AM1,                                  //      FLASH Data bus Bit 15 or Address A-1
        FL_ADDR,                                      //      FLASH Address bus 22 Bits
        FL_WE_N,                                      //      FLASH Write Enable
        FL_RST_N,                                     //      FLASH Reset
        FL_OE_N,                                      //      FLASH Output Enable
        FL_CE_N,                                      //      FLASH Chip Enable
        FL_WP_N,                                      //      FLASH Hardware Write Protect
        FL_BYTE_N,                                    //      FLASH Selects 8/16-bit mode
        FL_RY,                                        //      FLASH Ready/Busy
        //////////////////    GPIO    ////////////////////////////
        GPIO0_CLKIN,                                  //      GPIO Connection 0 Clock In Bus
        GPIO0_CLKOUT,                                 //      GPIO Connection 0 Clock Out Bus
        GPIO0_D,                                      //      GPIO Connection 0 Data Bus
        GPIO1_CLKIN,                                  //      GPIO Connection 1 Clock In Bus
        GPIO1_CLKOUT,                                 //      GPIO Connection 1 Clock Out Bus
        GPIO1_D                                       //      GPIO Connection 1 Data Bus
    );

//////////////////////        Clock Input             //////////////////////
input               CLOCK_50;                         //      50 MHz
input               CLOCK_50_2;                       //      50 MHz
//////////////////////        Push Button             //////////////////////
input   [2:0]   BUTTON;                               //      Pushbutton[2:0]
//////////////////////        DPDT Switch             //////////////////////
input   [9:0]   SW;                                   //      Toggle Switch[9:0]
//////////////////////        7-SEG Dispaly   //////////////////////
output  [6:0]   HEX0_D;                               //      Seven Segment Digit 0
output              HEX0_DP;                          //      Seven Segment Digit DP 0
output  [6:0]   HEX1_D;                               //      Seven Segment Digit 1
output              HEX1_DP;                          //      Seven Segment Digit DP 1
output  [6:0]   HEX2_D;                               //      Seven Segment Digit 2
output              HEX2_DP;                          //      Seven Segment Digit DP 2
output  [6:0]   HEX3_D;                               //      Seven Segment Digit 3
output              HEX3_DP;                          //      Seven Segment Digit DP 3
////////////////////////    LED             ////////////////////////
output  [63:0]  LEDG;                                 //      LED Green[9:0]
////////////////////////    UART    ////////////////////////
output              UART_TXD;                         //      UART Transmitter
input               UART_RXD;                         //      UART Receiver
output              UART_CTS;                         //      UART Clear To Send
input               UART_RTS;                         //      UART Request To Send
//////////////////////        SDRAM Interface ////////////////////////
inout   [15:0]  DRAM_DQ;                              //      SDRAM Data bus 16 Bits
output  [12:0]  DRAM_ADDR;                            //      SDRAM Address bus 13 Bits
output              DRAM_LDQM;                        //      SDRAM Low-byte Data Mask
output              DRAM_UDQM;                        //      SDRAM High-byte Data Mask
output              DRAM_WE_N;                        //      SDRAM Write Enable
output              DRAM_CAS_N;                       //      SDRAM Column Address Strobe
output              DRAM_RAS_N;                       //      SDRAM Row Address Strobe
output              DRAM_CS_N;                        //      SDRAM Chip Select
output              DRAM_BA_0;                        //      SDRAM Bank Address 0
output              DRAM_BA_1;                        //      SDRAM Bank Address 1
output              DRAM_CLK;                         //      SDRAM Clock
output              DRAM_CKE;                         //      SDRAM Clock Enable
//////////////////////        Flash Interface ////////////////////////
inout   [14:0]  FL_DQ;                                //      FLASH Data bus 15 Bits
inout               FL_DQ15_AM1;                      //      FLASH Data bus Bit 15 or Address A-1
output  [21:0]  FL_ADDR;                              //      FLASH Address bus 22 Bits
output              FL_WE_N;                          //      FLASH Write Enable
```

```
output          FL_RST_N;                       //      FLASH Reset
output          FL_OE_N;                        //      FLASH Output Enable
output          FL_CE_N;                        //      FLASH Chip Enable
output          FL_WP_N;                        //      FLASH Hardware Write Protect
output          FL_BYTE_N;                      //      FLASH Selects 8/16-bit mode
input           FL_RY;                          //      FLASH Ready/Busy
//////////////////////        GPIO    ////////////////////////////////
input   [1:0]   GPIO0_CLKIN;            //      GPIO Connection 0 Clock In Bus
output  [1:0]   GPIO0_CLKOUT;           //      GPIO Connection 0 Clock Out Bus
inout   [31:0]  GPIO0_D;                        //      GPIO Connection 0 Data Bus
input   [1:0]   GPIO1_CLKIN;            //      GPIO Connection 1 Clock In Bus
output  [1:0
]       GPIO1_CLKOUT;           //      GPIO Connection 1 Clock Out Bus
inout   [31:0]  GPIO1_D;                        //      GPIO Connection 1 Data Bus

//=============================================================
//  REG/WIRE declarations
//=============================================================
//      All inout port turn to tri-state
assign          DRAM_DQ         =       16'hzzzz;
assign          FL_DQ           =       16'hzzzz;
assign          GPIO0_D         =       32'hzzzzzzzz;
assign          GPIO1_D         =       32'hzzzzzzzz;

//////////////////////        7-SEG Display   ////////////////////////
wire [63:0] LED_reg; //these are the 64 registers whose values are sent to the LEDs
reg [25:0] counter; //this is the counter that is set to counter_max_value and then
//decremented every "tick" of the internal 50MHz reference clock
reg [25:0] phasecounter; //this is the counter that is used to control when the shift
//register gets shifted
reg shift_reg_flag; //this is the flag that is set every counter_max_value/100 cycles.
//whenever this flag gets set high, the shift register gets shifted

reg led_template_signal; //this is the output signal that is generated for the LEDs. It
//has the desired frequency and duty cycle.  This is the signal that is shifted through
//the shift register, allowing for different phase shifts
wire [25:0] counter_max_value; //this is the value that is used to generate the desired
//output frequency.  This value is set by changing the switches
wire [6:0] off_percentage; //this value is used to change the duty cycle of the LEDs. it
//is set by changing the switches
reg  [9:0] SW_reg1;
reg [9:0] SW_reg2; //these two registers are used to detect when the user has changed the
//switches
always @(posedge CLOCK_50 or negedge BUTTON[0])
begin
        SW_reg1 <= SW;
        SW_reg2 <= SW_reg1;
        if (!BUTTON[0])
        begin
                counter <= counter_max_value;
                led_template_signal <= 1'b0;
        end
        else
        begin
                if (counter == 26'd0)
                begin
                        counter <= counter_max_value;
                        led_template_signal <= 1'b0;
                end
                else if (counter > off_percentage*counter_max_value/100)
                begin
                        led_template_signal <= 1'b1;
                        counter <= counter -1'd1;
                end
                else
                begin
                        counter <= counter -1'd1;
                        led_template_signal <= 1'b0;
                end

                if (SW_reg1 != SW_reg2)
                begin
                        counter <= counter_max_value;
                end
        end
end

wire [25:0] shift_reg_flag_rate;
assign shift_reg_flag_rate = counter_max_value/100;

//phase block
always@(posedge CLOCK_50 or negedge BUTTON[0])
begin
        if (!BUTTON[0])
        begin
                phasecounter <= 26'd0;
                shift_reg_flag <= 1'b0;
        end
        else if(phasecounter == shift_reg_flag_rate - 1)
        begin
                shift_reg_flag <= 1;
                phasecounter <= 0;
        end
        else
        begin
                shift_reg_flag <= 0;
                phasecounter <= phasecounter + 1'd1;
        end
```

```
end


reg [99:0] shift_reg;



always @(posedge shift_reg_flag)
begin
        //fills up shift register
        shift_reg <= {shift_reg[98:0],led_template_signal};
end

assign counter_max_value =
(SW[3:0]==4'b0000)?
        26'd500000 : //max count=50M -> 50M/50M = 100Hz
(SW[3:0]==4'b0001)?
        26'd100000 : //max count=100k -> 50M/100K = 500Hz
(SW[3:0]==4'b0010)?
        26'd50000 : //max count=50k -> 50M/50k = 1kHz
(SW[3:0]==4'b0011)?
        26'd25000 : //max count=25k -> 50M/25k = 2kHz
(SW[3:0]==4'b0100)?
        26'd16666 : //max count=16.66k -> 50M/16.66k = 3kHz
(SW[3:0]==4'b0101)?
        26'd10000 : //max count=10k -> 50M/10k = 5kHz
(SW[3:0]==4'b0110)?
        26'd5000 : //max count=5k -> 50M/5k = 10kHz
(SW[3:0]==4'b0111)?
        26'd3333 ://max count=3.33k -> 50M/3.33k = 15kHz
(SW[3:0]==4'b1000)?
        26'd4000 : //max count=1250 -> 50M/4k = 12.5kHz
(SW[3:0]==4'b1001)?
        26'd4545: //max count=4545 -> 50M/4545 approx 11kHz
(SW[3:0]==4'b1010)?
        26'd3846 : //max count=3846 -> 50M/3846 = 13kHz
(SW[3:0]==4'b1011)?
        26'd1929 : //max count=330 -> 50M/4761 approx 10.5kHz(wrong)
(SW[3:0]==4'b1100)?
        26'd1933 : //max count=285 -> 50M/285 = 49.98kHz(wrong)
(SW[3:0]==4'b1101)?
        26'd1235 : //max count=250 -> 50M/250 = 38.46kHz(wrong)
(SW[3:0]==4'b1110)?
        26'd1250 : //max count=100 -> 50M/1250 = 40kHz
                26'd250;//max count=250 -> 50M/250=200kHz


assign off_percentage =
(SW[6:4]==3'b000)?
        7'd70 : //this gives a duty cycle of 30%
(SW[6:4]==3'b001)?
        7'd65 : //this gives a duty cycle of 35%
(SW[6:4]==3'b010)?
        7'd60 : //this gives a duty cycle of 40%
(SW[6:4]==3'b011)?
        7'd55 : //this gives a duty cycle of 45%
(SW[6:4]==3'b100)?
        7'd50 : //this gives a duty cycle of 50%
(SW[6:4]==3'b101)?
        7'd40 : //this gives a duty cycle of 60%
(SW[6:4]==3'b110)?
        7'd30 : //this gives a duty cycle of 70%
                7'd20;//this gives a duty cycle of 80%

//07/09 currently programmed for a focus of 0.02 m = 2 cm at 50 kHz and 10 kHz.

wire [8:0] output_bins;
assign output_bins = //[1:0]two final output bits for 2 possible values for LEDs to choose from, will be 4? in final 64
    ↪ design
(SW[8:7]==2'b00)?//40khz, f=7cm (with - sign)
        {shift_reg[85],shift_reg[21],shift_reg[76],shift_reg[53],shift_reg[5],shift_reg[80],shift_reg[54],shift_reg[27],
            ↪ shift_reg[0]}:
(SW[8:7]==2'b01)?//40khz f=7cm (WORKS?)
        {shift_reg[3],shift_reg[65],shift_reg[8],shift_reg[30],shift_reg[77],shift_reg[1],shift_reg[25],shift_reg[51],
            ↪ shift_reg[77]}:
        //
(SW[8:7]==2'b10)?//200kHz f=7cm try reversing them?
        {shift_reg[27],shift_reg[7],shift_reg[82],shift_reg[66],shift_reg[25],shift_reg[99],shift_reg[70],shift_reg[37],
            ↪ shift_reg[0]}:
              {shift_reg[73],shift_reg[93],shift_reg[18],shift_reg[34],shift_reg[75],shift_reg[1],shift_reg[30],
                  ↪ shift_reg[63],shift_reg[0]};
              //{shift_reg[44],shift_reg[24],shift_reg[99],shift_reg[83],shift_reg[42],shift_reg[16],shift_reg[87],
                  ↪ shift_reg[54],shift_reg[17]};
               //200 khz focus f = 7cm (v2 work?)

//Next, we assign the closest approximation we can make to a cylindrically symmetric set of
//rings, to use for simulating the actual lens phases.


//inner circle
assign {LED_reg[27],LED_reg[28],LED_reg[36],LED_reg[35]} = {4{output_bins[0]}};

//second circle
assign {LED_reg[19],LED_reg[20],LED_reg[29],LED_reg[37],LED_reg[43],LED_reg[44],LED_reg[26],LED_reg[34]} = {8{output_bins
    ↪ [1]}};

//third circle
```

```
assign {LED_reg[18],LED_reg[21],LED_reg[42],LED_reg[45]} = {4{output_bins[2]}};

//fourth circle
assign {LED_reg[11],LED_reg[12],LED_reg[33],LED_reg[25],LED_reg[51],LED_reg[52],LED_reg[30],LED_reg[38]} = {8{output_bins
    ↪ [3]}};

//fifth circle
assign {LED_reg[10],LED_reg[13],LED_reg[17],LED_reg[22],LED_reg[41],LED_reg[46],LED_reg[50],LED_reg[53]} = {8{output_bins
    ↪ [4]}};

//sixth circle
assign {LED_reg[3],LED_reg[4],LED_reg[9],LED_reg[14],LED_reg[24],
              LED_reg[32],LED_reg[31],LED_reg[39],LED_reg[49],LED_reg[54],LED_reg[59],LED_reg[60]} = {12{output_bins
                  ↪ [5]}};

//seventh circle
assign {LED_reg[58],LED_reg[61],LED_reg[40],LED_reg[16],LED_reg[23],LED_reg[47],LED_reg[2], LED_reg[5]} = {8{output_bins
    ↪ [6]}};

//eigth circle
assign {LED_reg[1],LED_reg[6],LED_reg[8],LED_reg[15],LED_reg[48],LED_reg[55],LED_reg[57],LED_reg[62]} =  {8{output_bins
    ↪ [7]}};


//ninth circle
assign {LED_reg[0],LED_reg[7],LED_reg[56],LED_reg[63]}={4{output_bins[8]}};

////////////////////////////////////////////////////////////////////////////////
//The Original ECE kids' ring assignments are below.
//
//the center square of the LED matrix
//assign {LED_reg[27],LED_reg[28],LED_reg[36],LED_reg[35]} = {4{output_bins[0]}};
//
//the second square of the LED matrix
//assign {LED_reg[18],LED_reg[19],LED_reg[20],LED_reg[21],
//              LED_reg[29],LED_reg[37],LED_reg[45],LED_reg[44],
//              LED_reg[43],LED_reg[42],LED_reg[34],LED_reg[26]} = {12{output_bins[1]}};
//
//the third square of the LED matrix
//assign {LED_reg[9],LED_reg[10],LED_reg[11],LED_reg[12],
//              LED_reg[13],LED_reg[14],LED_reg[22],LED_reg[30],
//              LED_reg[38],LED_reg[46],LED_reg[54],LED_reg[53],
//              LED_reg[52],LED_reg[51],LED_reg[50],LED_reg[49],
//              LED_reg[41],LED_reg[33],LED_reg[25],LED_reg[17]}= {20{output_bins[2]}};
//
//the outermost square of the LED matrix
//assign {LED_reg[0],LED_reg[1],LED_reg[2],LED_reg[3],
//              LED_reg[4],LED_reg[5],LED_reg[6],LED_reg[7],
//              LED_reg[15],LED_reg[23],LED_reg[31],LED_reg[39],
//              LED_reg[47],LED_reg[55],LED_reg[63],LED_reg[62],
//              LED_reg[61],LED_reg[60],LED_reg[59],LED_reg[58],
//              LED_reg[57],LED_reg[56],LED_reg[48],LED_reg[40],
//              LED_reg[32],LED_reg[24],LED_reg[16],LED_reg[8]}= {28{output_bins[3]}};
////////////////////////////////////////////////////////////////////////////////
assign HEX0_D = 7'b1111111;
assign HEX0_DP = 1'b0;
assign HEX1_D = 7'b1111111;
assign HEX1_DP = 1'b0;
assign HEX2_D = 7'b1111111;
assign HEX2_DP = 1'b0;
assign HEX3_D = 7'b1111111;
assign HEX3_DP = 1'b0;

assign LEDG[7] = LED_reg[0];
assign LEDG[0] = LED_reg[0];
assign LEDG[6] = LED_reg[9];
assign LEDG[1] = LED_reg[9];
assign LEDG[5] = LED_reg[18];
assign LEDG[2] = LED_reg[18];
assign LEDG[4:3] = {2{LED_reg[27]}};

//J4 ribbon output left side
//pin 1
//pin 3
assign GPIO0_D[2] = LED_reg[0];//pin 5
assign GPIO0_D[4] = LED_reg[1];//pin 7
assign GPIO0_D[6] = LED_reg[2];//pin 9
//5V pin 11
assign GPIO0_D[8] = LED_reg[3];//pin 13
assign GPIO0_D[10] = LED_reg[8];//pin 15
assign GPIO0_D[12] = LED_reg[9];//pin 17
assign GPIO0_CLKOUT[0] = LED_reg[10];//pin 19
assign GPIO0_CLKOUT[1] = LED_reg[11];//pin 21
assign GPIO0_D[16] = LED_reg[16];//pin 23
assign GPIO0_D[18] = LED_reg[17];//pin 25
assign GPIO0_D[20] = LED_reg[18];//pin 27
//3.3V pin 29
assign GPIO0_D[22] = LED_reg[19];//pin 31
assign GPIO0_D[24] = LED_reg[24];//pin 33
assign GPIO0_D[26] = LED_reg[25];//pin 35
assign GPIO0_D[28] = LED_reg[26];//pin 37
assign GPIO0_D[30] = LED_reg[27];//pin 39

//J4 ribbon output right side
assign GPIO0_D[0] = LED_reg[59];//pin 2
assign GPIO0_D[1] = LED_reg[58];//pin 4
assign GPIO0_D[3] = LED_reg[57];//pin 6
assign GPIO0_D[5] = LED_reg[56];//pin 8
```

```
assign GPIO0_D[7] = LED_reg[51];//pin 10
//GND pin 12
assign GPIO0_D[9] = LED_reg[50];//pin 14
assign GPIO0_D[11] = LED_reg[49];//pin 16
assign GPIO0_D[13] = LED_reg[48];//pin 18
assign GPIO0_D[14] = LED_reg[43];//pin 20
assign GPIO0_D[15] = LED_reg[42];//pin 22
assign GPIO0_D[17] = LED_reg[41];//pin 24
assign GPIO0_D[19] = LED_reg[40];//pin 26
assign GPIO0_D[21] = LED_reg[35];//pin 28
//GND pin 30
assign GPIO0_D[23] = LED_reg[34];//pin 32
assign GPIO0_D[25] = LED_reg[33];//pin 34
assign GPIO0_D[27] = LED_reg[32];//pin 36
//pin 38
//pin 40


//J5 ribbon output left side
//pin 1
//pin 3
assign GPIO1_D[2] = LED_reg[60];//pin 5
assign GPIO1_D[4] = LED_reg[61];//pin 7
assign GPIO1_D[6] = LED_reg[62];//pin 9
//5V pin 11
assign GPIO1_D[8] = LED_reg[63];//pin 13
assign GPIO1_D[10] = LED_reg[52];//pin 15
assign GPIO1_D[12] = LED_reg[53];//pin 17
assign GPIO1_D[29] = LED_reg[54];//pin 19
assign GPIO1_D[31] = LED_reg[55];//pin 21
assign GPIO1_D[16] = LED_reg[44];//pin 23
assign GPIO1_D[18] = LED_reg[45];//pin 25
assign GPIO1_D[20] = LED_reg[46];//pin 27
//3.3V pin 29
assign GPIO1_D[22] = LED_reg[47];//pin 31
assign GPIO1_D[24] = LED_reg[36];//pin 33
assign GPIO1_D[26] = LED_reg[37];//pin 35
assign GPIO1_D[28] = LED_reg[38];//pin 37
assign GPIO1_D[30] = LED_reg[39];//pin 39

//J5 ribbon output right side
assign GPIO1_D[0] = LED_reg[7];//pin 2
assign GPIO1_D[1] = LED_reg[6];//pin 4
assign GPIO1_D[3] = LED_reg[5];//pin 6
assign GPIO1_D[5] = LED_reg[4];//pin 8
assign GPIO1_D[7] = LED_reg[15];//pin 10
//GND pin 12
assign GPIO1_D[9] = LED_reg[14];//pin 14
assign GPIO1_D[11] = LED_reg[13];//pin 16
assign GPIO1_D[13] = LED_reg[12];//pin 18
assign GPIO1_D[14] = LED_reg[23];//pin 20
assign GPIO1_D[15] = LED_reg[22];//pin 22
assign GPIO1_D[17] = LED_reg[21];//pin 24
assign GPIO1_D[19] = LED_reg[20];//pin 26
assign GPIO1_D[21] = LED_reg[31];//pin 28
//GND pin 30
assign GPIO1_D[23] = LED_reg[30];//pin 32
assign GPIO1_D[25] = LED_reg[29];//pin 34
assign GPIO1_D[27] = LED_reg[28];//pin 36
//pin 38
//pin 40

endmodule
```

C.1.2 PhaseCalcModule.py

This file is a module that defines the calculations used for certain stock sound field patterns. The functions in this module are called in more recent versions of the code used to program the FPGA.

```
import math

def point(x=0, y=0, z=70, f=40000):
    """Creates a focus at a given point at the given frequency.

    This function accepts four inputs: three Cartesian coordinates given in mm,
    and a frequency given in Hz.  It will return a list of integers from 0 to
    99, representing the relative phase of each LED in an 8x8 array of LEDs
    with half an inch between each LED.

    The default input generates a focus at the origin a distance 70mm from the
    film at a frequency of 40000 Hz"""

    wavelength = 1000*343.0/f
    #The 1000 is to convert the speed of sound to millimeters
    kw=2*math.pi/wavelength
    pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]
    pointx = [ -44.45,  -31.75,  -19.05,   -6.35,  6.35, 19.05, 31.75, 44.45]
    r = []
    for i in range(len(pointy)):
        for j in range(len(pointx)):
            r.append(((x-pointx[j])**2 + (y-pointy[i])**2 + z**2)**.5)


    #we want to add a phase Phi to each LED such that kr+Phi is equivalent up
    #to an integer multiple of wavelengths for each LED.  We do this by
    #calculating kr mod 2pi and taking the negative of the result.  However,
    #due to the limited bin size of the FPGA, we must also
    #round the result to the nearest multiple of pi/50

    Phi=[]
    for element in r:
        q=kw*element
        w=q%(2*math.pi)
        t=50*w/math.pi
        y=round(t)
        Phi.append(-y%100)
    #now we convert the phases to integers, shift everything so that an LED of
    #choice has phase 0, then store the configuration in an array.
    psi=[]
    for i in range(len(Phi)):
        psi.append(int((Phi[i] - Phi[27])%100))
    return psi

def line(x=0, y=0, theta=math.pi/4, z=70, f=40000):
    """Creates a linear focus through the given point at given angle/frequency.

    The first two inputs determine a point that the line passes through;
    The next input determines the orientation of the line, and should be input
    in radians.  The last two inputs determine the focal plane and the
    frequency, respectively.All distances should be given in millimeters.  This
    function generates the phase configuration that is programmed into the
    FPGA to create the desired focus, and is also used in the code Simulate to
    simulate the desired focus.

    The default input creates a line with slope +1 through the origin 70 mm
    away from the film and at a frequency of 40000 Hz"""

    wavelength = 1000*343.0/f
    #The 1000 is to convert the speed of sound to millimeters
    kw=2*math.pi/wavelength
    pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]
    pointx = [ -44.45,  -31.75,  -19.05,   -6.35,  6.35, 19.05, 31.75, 44.45]
    rotatedy=[]
    x_0=[]
    y_0=[]
    for px in pointx:
        x_0.append(px-x)
    for py in pointy:
        y_0.append(py-y)
    #Then we rotate our coordinate system so that the line we want to produce
    #is the x-axis of the new system, and calculate the value of y for each LED
    for i in y_0:
        for j in x_0:
            rotatedy.append( -j*math.sin(theta) + i*math.cos(theta))
    ry = []
    for y in rotatedy:
        ry.append((y**2 + z**2)**.5)
    #We now calculate the phase offset needed to make all of the LEDs be the
    #same distance from the line (mod one wavelentgh), and apply that offset.
    #We also round the phase offset to within the bin size of the FPGA.
    Phi = []
    for element in ry:
        q = kw*element
        w = q%(2*math.pi)
        t=50*w/math.pi
        y = round(t)
        Phi.append(-y%100)
    psi=[]
    for i in range(len(Phi)):
```

```
        psi.append(int((Phi[i] - Phi[27])%100))
    return psi

def dospuntos(x1=0, y1=0, x2=0, y2=0, z=70, f=40000):
    """Creates two foci at chosen coordinates and frequency.

    The 8x8 array of LEDs can be split into two groups, much as they are on a
    chessboard.  This code causes one of these groups to create a focus at
    point 1, the coordinates are given by x1, y1, and z in millimeters, and the
    other to create a second focus at coordinates x2, y2, and z.  The final
    input, f, determines the frequency of the focus.

    If no input is provided, both groups will create a focus at the origin at a
    distance of 70mm from the films at 40000 Hz, providing identical results
    to the function points()."""
    wavelength = 1000*343.0/f
    #The 1000 is to convert the speed of sound to millimeters
    kw=2*math.pi/wavelength
    pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]
    pointx = [ -44.45,  -31.75,  -19.05,   -6.35,  6.35, 19.05, 31.75, 44.45]
    r=[]
    for j in range(len(pointy)):
        for i in range(len(pointx)):
            if i%2==j%2:
                r.append(((x1-pointx[i])**2 + (y1-pointy[j])**2 + z**2)**.5)
            else:
                r.append(((x2-pointx[i])**2 + (y2-pointy[j])**2 + z**2)**.5)

    Phi=[]
    for element in r:
        q=kw*element
        w=q%(2*math.pi)
        t=50*w/math.pi
        y=round(t)

    Phi.append(-y%100) #Sorry for poor notation, but hopefully this is readable
    psi=[]
    for i in range(len(Phi)):
        psi.append(int((Phi[i] - Phi[27])%100))
    return psi

def halfring(string="down", R=10, z=70, f=40000):
    """Creates a half-ring shape with a dot in its center.

    By default, the generated half-ring points downward (like a bowl or 'u') at
    a distance of 70mm from the films and at 40000 Hz.  One can make this
    shape point up, left, or right by inputting the corresponding direction
    as a string"""

    wavelength = 1000*343.0/f
    #The 1000 is to convert the speed of sound to millimeters
    kw=2*math.pi/wavelength
    pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]
    pointx = [ -44.45,  -31.75,  -19.05,   -6.35,  6.35, 19.05, 31.75, 44.45]
    r=[]

    if string in ["up", "UP", "u", "U", "Up", "uP"]:
        for j in range(len(pointy)):
            for i in range(len(pointx)):
                theta=(math.atan(pointy[j]/pointx[i]))
                if theta>0:
                    theta-=math.pi
                r.append(((pointx[i]-R*math.cos(theta))**2 +
                (pointy[j]- R*math.sin(theta))**2+z**2)**.5)
    elif string in ["left", "Left", "LEFT", "L", "l", "lEFT"]:
        for j in range(len(pointy)):
            for i in range(len(pointx)):
                theta=(math.atan(pointy[j]/pointx[i]))
                if theta>0:
                    theta-=math.pi
                r.append(((pointx[i]-R*math.cos(theta))**2 +
                (pointy[j]- R*math.sin(theta))**2+z**2)**.5)
    elif string in ["right", "Right", "RIGHT", "R", "r", "rIGHT"]:
        for j in range(len(pointy)):
            for i in range(len(pointx)):
                theta=(math.atan(pointy[j]/pointx[i]))
                r.append(((pointx[i]-R*math.cos(theta))**2 +
                (pointy[j]- R*math.sin(theta))**2+z**2)**.5)
    elif string not in ["down", "Down", "DOWN", "d", "D", "dOWN"]:
        print ("Invalid input: 'string' should be either 'up', 'down', 'left',"
        "or 'right'.  The default input of 'down' will be assumed.")
        string="down"
    if string in ["down", "Down", "DOWN", "D", "d", "dOWN"]:
        for j in range(len(pointy)):
            for i in range(len(pointx)):
                theta=(math.atan(pointy[j]/pointx[i]))
                if theta>0:
                    theta-=math.pi
                r.append(((pointx[i]-R*math.cos(theta))**2 +
                (pointy[j]- R*math.sin(theta))**2+z**2)**.5)
    Phi=[]
    for element in r:
        q=kw*element
        w=q%(2*math.pi)
        t=50*w/math.pi
        y=round(t)

        Phi.append(-y%100)
```

```
    psi=[]
    for i in range(len(Phi)):
        psi.append(int((Phi[i] - Phi[27])%100))
    return psi

def ring(n=8.575, alpha= math.pi/4,  z=70, f=40000):
    """At the moment, I do not recommend inputting anything for this function,
    as the variables listed alter the resulting focus unpredictably."""

    wavelength = 1000*343.0/f
    #The 1000 is to convert the speed of sound to millimeters
    kw=2*math.pi/wavelength
    pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]
    pointx = [ -44.45,  -31.75,  -19.05,   -6.35,  6.35, 19.05, 31.75, 44.45]
    r=[]
    for i in range(len(pointx)):
        for j in range(len(pointx)):
            r.append(((pointx[j])**2 + (pointy[i])**2)**.5)
    Phi=[]
    for element in r:
        phasedifference=-(n-1)*kw*element*math.tan(alpha)
        Phi.append(round(phasedifference*50.0/math.pi)%100)
    psi=[]
    for i in range(len(Phi)):
        psi.append(int((Phi[i] - Phi[27])%100))
    return psi
```

C.1.3 FPGAControl.py

This program allows one to specify a single location and frequency and creates a sound "focus" of the appropriate frequency at that location. The location is changed by modifiying the values of x, y, and z in the program before running it; the origin of this coordinate system corresponds to the center of the phased array, with z indicating distance from the phased array and y corresponding to vertical displacement. Likewise, changing f before running the program changes the frequency. Running this program creates a new version of the file phased_array_control_backup.v, called phased_array_control.v, which will generate the desired sound field once it has been compiled and uploaded to the FPGA through the Quartus software.

```
import math

#Insert the desired coordinates of the focus (in mm) and the frequency (in Hz) below:
x=0.0
y=0.0
z=70000000000000.0
f=40000.0

#First we generate an array of the distances of each LED from the desired focus.
r = []
pointy = [-44.45, -31.75, -19.05, -6.35, 6.35, 19.05, 31.75, 44.45]
pointx = [44.45, 31.75, 19.05, 6.35, -6.35, -19.05, -31.75, -44.45]
for i in range(len(pointy)):
    for j in range(len(pointx)):
        r.append(((x-pointx[j])**2 + (y-pointy[i])**2 + z**2)**.5) #the plus in the y argument is equivalent to starting
            ↪ with positive y and going down.

#Now we calculate the wavelength of the sound.
wavelength = 1000*343.0/f #The 1000 is to convert the speed of sound to millimeters
k=2*(math.pi)/wavelength
#we want to add a phase Phi to each LED such that kr+Phi is equivalent up to an integer
#multiple of wavelengths for each LED.  We do this by calculating kr mod 2pi and taking the
#negative of the result.  However, due to the limited bin size of the FPGA, we must also
#round the result to the nearest multiple of pi/50

Phi=[]
for element in r:
    q=k*element
    w=q%(2*(math.pi))
    t=50*w/(math.pi)
    y=round(t)

    Phi.append(-y%100) #Sorry for poor notation, but hopefully this is readable

psi=[]
for i in range(len(Phi)):
    psi.append((Phi[i] - Phi[27])%100)

#Now all that's left is to output the phases to the FPGA:
#Well, technically we should first change the output from floating point to integer:
Psi = []
for i in range(len(psi)):
    Psi.append(int(psi[i]))


f=open('phased_array_control_backup.v')
text=f.readlines()

f.close()
g=open('phased_array_control.v', 'w')



phases= "{"

for i in range(64):
    phases+="shift_reg["
    phases+=str(Psi[i])
    if i<63:
        phases+="],"
    else:
        phases+="]}"
Output=[]
for i in range (309):
    Output.append(text[i])
Output.append("wire [63:0] output_bins;\n")
Output.append("assign output_bins =\n")
Output.append("(SW[8:7] == 2'b00)?\n")
Output.append("    "+phases+":\n")
Output.append("(SW[8:7] == 2'b01)?\n")
Output.append("    "+phases+":\n")
Output.append("(SW[8:7] == 2'b10)?\n")
Output.append("    "+phases+":\n")
Output.append("    "+"    "+phases+";\n")
for i in range (64):
    assign=""
    assign+="assign{LED_reg["
    assign+=str(i)
    assign+="] = {1{output_bins["
    assign +=str(i)
```

```
        assign+="]}};\n"
        Output.append(assign)
    for i in range (490-356):
        Output.append(text[i+356])
    for i in range(len(Output)):
        g.write(Output[i])
    g.close()
```

C.1.4 FPGAControlLine.py

Similarly to FPGAControl.py, this program allows one to generate a linear sound field by entering the appropriate parameters into the program before running it.

```
import math

#Insert the desired angle of the linear focus here (theta=0 corresponds to a horizontal line; use radians)
theta=math.pi/4.0
#Insert the distance from the films to the plane of observation (in mm; note that all distances will be given in
    ↪ millimeters in this code):
z = 70
#Insert the frequency:
f=40000
#first we generate a list of possible values of x and y
y_0 = [-44.45, -31.75, -19.05, -6.35, 6.35, 19.05, 31.75, 44.45]
x_0=[]
for y in y_0:
        v=-y
        x_0.append(v)
#Then we rotate our coordinate system so that the line we want to produce is the x-axis of the new system, and calculate
    ↪ the value of y for each LED:
rotatedy=[]
for i in y_0:
    for j in x_0:
        rotatedy.append( -j*math.sin(theta) + i*math.cos(theta))
#now we calculate the minimum distance of each LED from the line focus:
wavelength = 1000*343/f
k = 2*math.pi/wavelength
ry = []
for y in rotatedy:
    ry.append((y**2 + z**2)**.5)
#We now calculate the phase offset needed to make all of the LEDs be the same distance from the line (mod one wavelentgh)
    ↪ , and apply that offset. We also round the phase offset to within the bin size of the FPGA.
psi = []
for element in ry:
    q = k*element
    w = q%(2*math.pi)
    t=50*w/math.pi
    y = round(t)
    psi.append(-y%100)

#All that's left is to convert the phases to integers and then output them to the verilog file that runs the FPGA:
Psi = []
for i in range(len(psi)):
    Psi.append(int(psi[i]))
f=open('phased_array_control_backup.v')
text=f.readlines()

f.close()
g=open('phased_array_control.v', 'w')



phases= "{"

for i in range(64):
    phases+="shift_reg["
    phases+=str(Psi[i])
    if i<63:
        phases+="],"
    else:
        phases+="]}"
Output=[]
for i in range (309):
    Output.append(text[i])
Output.append("wire␣[63:0]␣output_bins;\n")
Output.append("assign␣output_bins␣=\n")
Output.append("(SW[8:7]␣==␣2'b00)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("(SW[8:7]␣==␣2'b01)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("(SW[8:7]␣==␣2'b10)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("␣␣␣␣"+"␣␣␣␣"+phases+";\n")
for i in range (64):
    assign=""
    assign+="assign{LED_reg["
    assign+=str(i)
    assign+="]}␣=␣{1{output_bins["
    assign +=str(i)
    assign+="]}};\n"
    Output.append(assign)
for i in range (490-356):
    Output.append(text[i+356])
for i in range(len(Output)):
    g.write(Output[i])
g.close()
```

C.1.5 FPGA2Dots.py

This program allows one to generate sound "foci" at two specified points at the same frequency using the spatial-multiplexing technique. The method of operation is similar to that of FPGAControl.py.

```
import math
#enter a focal plane, frequency, and two sets of coordinates; for testing purposes, it is preferable that the two points
    ↪ be on opposite sides of the origin.
z=70
f=40000
x1=-10
y1=0
x2=10
y2=0
#Input simulated scan coordinates and number of bins
xmin = -15
xmax = 15
ymin=-15
ymax=15
numx=31
numy=31
#I just want to see if this works: first, we list out all possible coordinates:
pointx = [44.45, 31.75, 19.05, 6.35, -6.35, -19.05, -31.75, -44.45]
pointy=[]
for x in pointx:
    pointy.append(-x)
#now we split the board into two groups in a checkerboard pattern. One group will create a focus at point 1, and the
    ↪ other at point 2
r=[]
for j in range(len(pointy)):
    for i in range(len(pointx)):
            if i%2==j%2:
                r.append(((x1-pointx[i])**2 + (y1-pointy[j])**2 + z**2)**.5)
            else:
                r.append(((x2-pointx[i])**2 + (y2-pointy[j])**2 + z**2)**.5)
#Now we copy-paste some stuff from my original simulation code:
wavelength = 1000*343.0/f #The 1000 is to convert the speed of sound to millimeters
k=2*math.pi/wavelength
#we want to add a phase Phi to each LED such that kr+Phi is equivalent up to an integer
#multiple of wavelengths for each LED.  We do this by calculating kr mod 2pi and taking the
#negative of the result.  However, due to the limited bin size of the FPGA, we must also
#round the result to the nearest multiple of pi/50

Phi=[]
for element in r:
    q=k*element
    w=q%(2*math.pi)
    t=50*w/math.pi
    y=round(t)

    Phi.append(-y%100) #Sorry for poor notation, but hopefully this is readable

psi=[]
for i in range(len(Phi)):
    psi.append((Phi[i] - Phi[27])%100)
Psi = []
for i in range(len(psi)):
    Psi.append(int(psi[i]))


f=open('phased_array_control_backup.v')
text=f.readlines()

f.close()
g=open('phased_array_control.v', 'w')



phases= "{"

for i in range(64):
    phases+="shift_reg["
    phases+=str(Psi[i])
    if i<63:
        phases+="],"
    else:
        phases+="]}"
Output=[]
for i in range (309):
    Output.append(text[i])
Output.append("wire [63:0] output_bins;\n")
Output.append("assign output_bins =\n")
Output.append("(SW[8:7] == 2'b00)?\n")
Output.append("    "+phases+":\n")
Output.append("(SW[8:7] == 2'b01)?\n")
Output.append("    "+phases+":\n")
Output.append("(SW[8:7] == 2'b10)?\n")
Output.append("    "+phases+":\n")
Output.append("    "+"    "+phases+";\n")
for i in range (64):
    assign=""
    assign+="assign{LED_reg["
    assign+=str(i)
    assign+="]} = {1{output_bins["
    assign +=str(i)
```

```
        assign+="]}};\n"
        Output.append(assign)
    for i in range (490-356):
        Output.append(text[i+356])
    for i in range(len(Output)):
        g.write(Output[i])
    g.close()
```

C.1.6 FPGASemicircle.py

This program allows one to generate a semicircular sound intensity pattern.

```
import math
#Note: the following code's main purpose is to give a color plot of the intensity of a sound focus given an arbitrary
    ↪ array of phases psi, a frequency, and a focal plane.  As I will likely need to do this for many different
#types of foci, I recommend that I copy-paste code from the (Output to FPGA) code to this code in order to generate Psi
    ↪ for the needed focus.  The current code is for a ring.
#Insert the desired coordinates of the focus (in mm) and the frequency (in Hz) below:
x1=-7.5
y1=7.5
x2=7.5
y2=7.5
z=70.0
f=40000.0
R=10.0
epsilon=4
#Input simulated scan coordinates and number of bins
xmin = -15
xmax = 15
ymin=-15
ymax=15
numx=31
numy=31
#First we generate an array of the distances of each LED from the central axis
r = []
pointy = [-44.45, -31.75, -19.05, -6.35, 6.35, 19.05, 31.75, 44.45]
pointx = []
for y in pointy:
    pointx.append(-y)
for j in range(len(pointy)):
    for i in range(len(pointx)):

       # if j<5 and i%2==j%2:
        #    r.append(((pointx[i]-x2)**2 + (pointy[j]-y2)**2 + z**2)**.5)
        #elif j<5 and (i+1)%2==j%2:
        #    r.append(((pointx[i]-x1)**2 + (pointy[j]-y1)**2 + z**2)**.5)


        #else:

            theta=(math.atan(pointy[j]/pointx[i]))
            if theta>0:
                theta-=math.pi
            r.append(((pointx[i]-R*math.cos(theta))**2 + (pointy[j]- R*math.sin(theta))**2+z**2)**.5)

#Now we calculate the wavelength of the sound.
wavelength = 1000*343.0/f #The 1000 is to convert the speed of sound to millimeters
k=2*math.pi/wavelength
#we want to add a phase Phi to each LED such that kr+Phi is equivalent up to an integer
#multiple of wavelengths for each LED.  We do this by calculating kr mod 2pi and taking the
#negative of the result.  However, due to the limited bin size of the FPGA, we must also
#round the result to the nearest multiple of pi/50

Phi=[]
for element in r:
    q=k*element
    w=q%(2*math.pi)
    t=50*w/math.pi
    y=round(t)

    Phi.append(-y%100) #Sorry for poor notation, but hopefully this is readable

psi=[]
for i in range(len(Phi)):
    if i==1%2:
        psi.append((Phi[i] +50 - Phi[27])%100)
    else:
        psi.append((Phi[i] - Phi[27])%100)


psi=[]
for i in range(len(Phi)):
    psi.append((Phi[i] - Phi[27])%100)
Psi=[]
#All that's left is to convert the phases to integers and then output them to the verilog file that runs the FPGA:
for i in range(len(psi)):
    Psi.append(int(psi[i]))
f=open('phased_array_control_backup.v')
text=f.readlines()

f.close()
g=open('phased_array_control.v', 'w')

phases= "{"

for i in range(64):
    phases+="shift_reg["
    phases+=str(Psi[i])
    if i<63:
        phases+="],"
    else:
        phases+="]}"
Output=[]
for i in range (309):
```

```
    Output.append(text[i])
Output.append("wire␣[63:0]␣output_bins;\n")
Output.append("assign␣output_bins␣=\n")
Output.append("(SW[8:7]␣==␣2'b00)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("(SW[8:7]␣==␣2'b01)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("(SW[8:7]␣==␣2'b10)?\n")
Output.append("␣␣␣␣"+phases+":\n")
Output.append("␣␣␣␣"+"␣␣␣␣"+phases+";\n")
for i in range (64):
    assign=""
    assign+="assign{LED_reg["
    assign+=str(i)
    assign+="]}␣=␣{1{output_bins["
    assign +=str(i)
    assign+="]}};\n"
    Output.append(assign)
for i in range (490-356):
    Output.append(text[i+356])
for i in range(len(Output)):
    g.write(Output[i])
g.close()
```

C.1.7 FPGASmile.py

This program uses the time-multiplexing technique to generate a sound intensity pattern resembling a smiling face.

```
import math
#First, we enter information about the entire focus:
z=70.0
f=40000.0
#we also need some information about how fast we want to switch between the eyes and the mouth, and for how long to hold
    ↪ each:
cycles=8
eye_cycles=4
mouth_cycles=cycles-eye_cycles
#now we enter information about the eyes:
x1=-7.5
x2=7.5
y1=7.5
y2=7.5
#and now for the mouth:
R = 10.0

#now we calculate the two required phase configurations, again starting with the eyes:

pointy = [ 44.45,  31.75,  19.05,   6.35,  -6.35, -19.05, -31.75, -44.45]

pointx=[]
for y in pointy:
    pointx.append(-y)
#now we split the board into two groups in a checkerboard pattern. One group will create a focus at point 1, and the
    ↪ other at point 2
r_eyes=[]
for j in range(len(pointy)):
    for i in range(len(pointx)):
        if j==4 and i==2:
            r_eyes.append(((x2-pointx[i])**2 + (y2-pointy[j])**2 + z**2)**.5)
        elif j==3 and i==2:
           r_eyes.append(((x1-pointx[i])**2 + (y1-pointy[j])**2 + z**2)**.5)

        elif i%2==j%2:
            r_eyes.append(((x1-pointx[i])**2 + (y1-pointy[j])**2 + z**2)**.5)
        else:
            r_eyes.append(((x2-pointx[i])**2 + (y2-pointy[j])**2 + z**2)**.5)
#Now we copy-paste some stuff from my original simulation code:
wavelength = 1000*343.0/f #The 1000 is to convert the speed of sound to millimeters
k=2*math.pi/wavelength
#we want to add a phase Phi to each LED such that kr+Phi is equivalent up to an integer
#multiple of wavelengths for each LED.  We do this by calculating kr mod 2pi and taking the
#negative of the result.  However, due to the limited bin size of the FPGA, we must also
#round the result to the nearest multiple of pi/50

Phi_eyes=[]
for element in r_eyes:
    q=k*element
    w=q%(2*math.pi)
    t=50*w/math.pi
    y=round(t)

    Phi_eyes.append(-y%100) #Sorry for poor notation, but hopefully this is readable

psi_eyes=[]
for i in range(len(Phi_eyes)):
    psi_eyes.append((Phi_eyes[i] - Phi_eyes[27])%100)

#now we calculate the mouth phases:
r_mouth = []

for j in range(len(pointy)):
    for i in range(len(pointx)):

            theta=(math.atan(pointy[j]/pointx[i]))
            if theta>0:
                theta-=math.pi
            r_mouth.append(((pointx[i]-R*math.cos(theta))**2 + (pointy[j]- R*math.sin(theta))**2+z**2)**.5)


Phi_mouth=[]
for element in r_mouth:
    q=k*element
    w=q%(2*math.pi)
    t=50*w/math.pi
    y=round(t)

    Phi_mouth.append(-y%100) #Sorry for poor notation, but hopefully this is readable

psi_mouth=[]
for i in range(len(Phi_mouth)):
    if i==1%2:
        psi_mouth.append((Phi_mouth[i] - Phi_mouth[27])%100)
    else:
        psi_mouth.append((Phi_mouth[i] - Phi_mouth[27])%100)

#and now for the actual file-writing part:
#Well, technically we should first change the output from floating point to integer:
Psi_eyes = []
for i in range(len(psi_eyes)):
    Psi_eyes.append(int(psi_eyes[i]))
```

```
Psi_mouth = []
for i in range(len(psi_mouth)):
    Psi_mouth.append(int(psi_mouth[i]))

f=open('phased_array_control_backup.v')
text=f.readlines()

f.close()
g=open('phased_array_control.v', 'w')



phases_eyes= "{"

for i in range(64):
    phases_eyes+="shift_reg["
    phases_eyes+=str(Psi_eyes[i])
    if i<63:
        phases_eyes+="],"
    else:
        phases_eyes+="}\n"

phases_mouth= "{"

for i in range(64):
    phases_mouth+="shift_reg["
    phases_mouth+=str(Psi_mouth[i])
    if i<63:
        phases_mouth+="],"
    else:
        phases_mouth+="}\n"

Output=[]
for i in range (172):
    Output.append(text[i])
Output.append("reg [25:0] metacounter; \n")
for i in range(172, 182):
    Output.append(text[i])
Output.append("wire [25:0] metacounter_max_value; \n")
for i in range (182, 198):
        Output.append(text[i])
Output.append("        "+"if (counter == 26'd0) ")
Output.append("        "+"begin\n")
Output.append("            "+"if (metacounter == 26'd0)\n")
Output.append("            "+"begin\n")
Output.append("                "+"counter <= counter_max_value;\n")
Output.append("                "+"metacounter <= metacounter_max_value;\n")
Output.append("                "+"led_template_signal <=1'b0;\n")
Output.append("            "+"end\n")
Output.append("            "+"else\n")
Output.append("            "+"begin\n")
Output.append("                "+"counter <= counter_max_value;\n")
Output.append("                "+"metacounter <= metacounter-1'd1;\n")
Output.append("                "+"led_template_signal <= 1'b0;\n")
Output.append("            "+"end\n")
Output.append("        "+"end\n")
for i in range(203, 288):
    Output.append(text[i])
Output.append("assign metacounter_max_value = \n")
Output.append("7'd"+str((cycles-1)) + ";\n")
for i in range (288, 309):
    Output.append(text[i])
Output.append("wire [63:0] output_bins;\n")
Output.append("assign output_bins=\n")
Output.append("(SW[8:7] == 2'b00 && metacounter<"+ str(eye_cycles) + ")?\n")
Output.append("    "+str(phases_eyes)+":\n")
Output.append("(SW[8:7] == 2'b01 && metacounter<"+ str(eye_cycles) + ")?\n")
Output.append("    "+str(phases_eyes)+":\n")
Output.append("(SW[8:7] == 2'b10 && metacounter<" + str(eye_cycles) + ")?\n")
Output.append("    "+str(phases_eyes)+":\n")
Output.append("(SW[8:7] == 2'b11 && metacounter<" + str(eye_cycles) + ")?\n")
Output.append("    "+str(phases_eyes)+":\n")
Output.append("(SW[8:7] == 2'b00 && metacounter>" + (str(eye_cycles-1)) + ")?\n")
Output.append("    "+str(phases_mouth)+":\n")
Output.append("(SW[8:7] == 2'b01 && metacounter>" + str((eye_cycles-1)) + ")?\n")
Output.append("    "+str(phases_mouth)+":\n")
Output.append("(SW[8:7] == 2'b10 && metacounter>" + str((eye_cycles-1)) + ")?\n")
Output.append("    "+str(phases_mouth)+":\n")
Output.append("    "+"    "+str(phases_mouth)+";\n")
for i in range (64):
    assign=""
    assign+="assign{LED_reg["
    assign+=str(i)
    assign+="] = {1{output_bins["
    assign +=str(i)
    assign+="]}};\n"
    Output.append(assign)
for i in range (490-356):
    Output.append(text[i+356])
for i in range(len(Output)):
    g.write(Output[i])
g.close()
```

C.1.8 FPGAAny2.py

This program is the most versatile FPGA program, although it is thus also the most complex. The program allows you to specify what shape of focus you would like to produce, and allows one to simultaneously take advantage of both spatial multiplexing and time multiplexing. This program also allows you to specify two different frequencies for the separate spatial groups, which makes this code useful for the nonlinear experiments described in Chapter 4. The older versions of this program, FPGAAny and FPGAAny3, can more or less be regarded as deprecated.

It should also be noted that it's possible that there is a bug in the program which manifests when one of the frequencies used is not a factor of the clock rate, 50 MHz; this issue may be worth investigating.

```
import math
import PhaseCalcModule as phaser
import numpy as np
#please input a list of the TYPES of objects you would like, according to the
#following code (I'm not going to comment the legend though;
#that way, you can type the word used or the number for the same results).
dot=0
line=1
two_dots=2
ring=3
half_ring=4

################################################################################
                        #FIRST INPUT: OBJECT TYPES
TYPES1=[]
TYPES2=[]
#The LEDs will be split into two groups in a chessboard pattern.
#The indices 1 and 2 represent the separate coding of these two groups.

#setting 1: 00
TYPES1.append([0])
TYPES2.append([0])
#setting 2: 01
TYPES1.append([0])
TYPES2.append([0])
#setting 3: 10
TYPES1.append([0])
TYPES2.append([0])
#setting 4: 11
TYPES1.append([0])
TYPES2.append([0])

################################################################################

#We now need a list of descriptions of the items you would like to create.
#Details below.

#For a DOT, the description is the coordinates of the point, in the format
#[x, y]. Note that all distances should be expressed in millimeters.

#For a LINE, the description is an arbitrary point on the line and an angle,
#in the format [x, y, theta]; however, x and y input has not yet been tested.

#For TWO DOTS (generated simultaneously), the description is the coordinates of
#each point, in the format [x1, y1, x2, y2]

#For a RING, input the index of refraction and angle of an appropriate axicon.
#I recommend [8.575, math.pi/4]

#Descriptions for HALF-RINGS are limited at the moment; enter "up", "down",
#"left", or "right"(including the quotes) to indicate half-ring direction
#(For example, "down" produces a half-ring somewhat like a bowl or a u;
#"left" produces a C-like half-ring.)

################################################################################
                        #SECOND INPUT: DESCRIPTIONS
coordinates1=[]
coordinates2=[]
#Setting 1
coordinates1.append([(0,0)])
coordinates2.append([(0,0)])
#Setting 2
coordinates1.append([(0,0)])
coordinates2.append([(0, 0)])
#Setting 3
coordinates1.append([(0,0)])
coordinates2.append([(0, 0)])
#Setting 4
coordinates1.append([(0,0)])
coordinates2.append([(0, 0)])
################################################################################

#Please also specify a focal plane and the frequenc(y/ies)

################################################################################
z=70                #THIRD INPUT: FOCAL PLANE
```

```
f1=40000                        #FOURTH INPUT: FREQUENCIES
f2=50000
###############################################################################
#Please specify the number of cycles you would like to hold each point for.
#Note that this input can either be a single number, in which case all points
#on the curve will be held for an equal amount of time,
#or a 1-D array of the same length as "coordinates", which allows you to place
#more weight on certain points than others.

#Note: None of the entries of this array should be zero.  If you want a focus
#to be held for 0 cycles, just don't enter it at all.  Ignoring this may
#interfere with the code, although I have not tested this yet.

###############################################################################
cycles_per_point1 = [4,4,4,4]              #FINAL INPUT
cycles_per_point2 = [4,4,4,4]
###############################################################################









###############################################################################


#We start by loading the base file, which contains most of the code that we
#need to generate the focus:
file=open('phased_array_control_backup.v')
text=file.readlines()
file.close()
###############################################################################

#Now we will start constructing the output code using the base file and the
#input above.
Output = []
#The first portion of the code works fine on its own, and will be copied here.

for i in range(170):
    Output.append(text[i])
#We now create a bunch of different objects we will need.
Output.append(
"reg[25:0] counter1; \n"+
"reg[25:0] counter2; \n"+
#We now create an object that I call the metacounter.  This objects counts the
#number of times the object "counter" has reached zero (it counts the counter,
#hence metacounter)
"reg [25:0] metacounter1; \n"+
#We'll need a second one of these.
"reg [25:0] metacounter2; \n"+
"reg [25:0] phasecounter1; \n"+
"reg [25:0] phasecounter2; \n"+
"reg shift_reg_flag1; \n"+
"reg shift_reg_flag2; \n"+
"reg led_template_signal1; \n"+
"reg led_template_signal2; \n"+
"wire [25:0] counter1_max_value; \n"+
"wire [25:0] counter2_max_value; \n"+
"wire [25:0] metacounter1_max_value; \n"+
"wire [25:0] metacounter2_max_value; \n")
###############################################################################

#Most of the next portion of code is fine, but we have to rewrite the portion
#handling what happens when counter reaches zero.  We split this case into two
#cases, one for if the metacounter is also zero, and one for if the metacounter
#is not zero


for i in range(182, 193):
        Output.append(text[i])
Output.append(
"        counter1 <= counter1_max_value;\n"+
"        counter2 <= counter2_max_value;\n"+
"        led_template_signal1 <=1'b0;\n"+
"        led_template_signal2 <=1'b0;\n"+
"    end\n"+
"    else\n"+
"    begin\n"+
"        if (counter1 == 26'd0) \n"+
"        begin\n"+
"            if (metacounter1 == 26'd0)\n"+
"            begin\n"+
"                counter1 <= counter1_max_value;\n"+
"                metacounter1 <= metacounter1_max_value;\n"+
"                led_template_signal1 <=1'b0;\n"+
"            end\n"+
"            else\n"+
"            begin\n"+
"                counter1 <= counter1_max_value;\n"+
"                metacounter1 <= metacounter1 -1'd1;\n"+
"                led_template_signal1 <= 1'b0;\n"+
"            end\n"+
"        end\n"+
"        else if (counter1 > off_percentage*counter1_max_value/100) \n"+
"        begin \n"+
```

```
"␣␣␣␣␣␣␣␣␣␣␣led_template_signal1␣<=␣1'b1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣counter1␣<=counter1␣-␣1'd1;␣\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣else\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣counter1␣<=␣counter1␣-␣1'd1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣led_template_signal1␣<=␣1'b0;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣if␣(counter2␣==␣26'd0)␣\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣if␣(metacounter2␣==␣26'd0)\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter2␣<=␣counter2_max_value;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣metacounter2␣<=␣metacounter2_max_value;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣led_template_signal2 <=1'b0;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣else\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter2␣<=␣counter2_max_value;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣metacounter2␣<=␣metacounter2-1'd1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣led_template_signal2␣<=␣1'b0;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣else␣if␣(counter2␣>␣off_percentage*counter2_max_value/100)␣\n"+
"␣␣␣␣␣␣␣␣begin␣\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣led_template_signal2␣<=␣1'b1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣counter2␣<=counter2␣-␣1'd1;␣\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣else\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣counter2␣<=␣counter2␣-␣1'd1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣led_template_signal2␣<=␣1'b0;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣if␣(SW_reg1␣!=␣SW_reg2)\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣counter1␣<=␣counter1_max_value;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣counter2␣<=␣counter2_max_value;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣end\n"+
"end\n"+
"\n"+
"wire␣[25:0]␣shift_reg_flag_rate1;␣\n"+
"assign␣shift_reg_flag_rate1␣=␣counter1_max_value/100;\n"+
"wire␣[25:0]␣shift_reg_flag_rate2;␣\n"+
"assign␣shift_reg_flag_rate2␣=␣counter2_max_value/100;\n"+
"\n"+
"//phase␣block\n"+
"always@(posedge␣CLOCK_50␣or␣negedge␣BUTTON[0])\n"+
"begin\n"+
"␣␣␣␣if␣(!BUTTON[0])\n"+
"␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣phasecounter1␣<=␣26'd0;\n"+
"␣␣␣␣␣␣␣␣phasecounter2␣<=␣26'd0;\n"+
"␣␣␣␣␣␣␣␣shift_reg_flag1␣<=␣1'b0;\n"+
"␣␣␣␣␣␣␣␣shift_reg_flag2␣<=␣1'b0;\n"+
"␣␣␣␣end\n"+
"␣␣␣␣else\n"+
"␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣if␣(phasecounter1␣==␣shift_reg_flag_rate1-1)\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣shift_reg_flag1␣<=␣1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣phasecounter1␣<=0;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣else\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣shift_reg_flag1␣<=0;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣phasecounter1␣<=␣phasecounter1␣+␣1'd1;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣if␣(phasecounter2␣==␣shift_reg_flag_rate2-1)\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣shift_reg_flag2␣<=␣1;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣phasecounter2␣<=0;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣␣␣␣␣else\n"+
"␣␣␣␣␣␣␣␣begin\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣shift_reg_flag2␣<=0;\n"+
"␣␣␣␣␣␣␣␣␣␣␣␣phasecounter2␣<=␣phasecounter2␣+␣1'd1;\n"+
"␣␣␣␣␣␣␣␣end\n"+
"␣␣␣␣end\n"+
"end\n"+
"\n"+
"\n"+
"reg␣[99:0]␣shift_reg1;\n"+
"always␣@(posedge␣shift_reg_flag1)\n"+
"begin\n"+
"␣␣␣␣//fills␣up␣shift␣register\n"+
"␣␣␣␣shift_reg1␣<=␣{shift_reg1[98:0],␣led_template_signal1};\n"+
"end\n"+
"reg␣[99:0]␣shift_reg2;␣\n"+
"always␣@(posedge␣shift_reg_flag2)\n"+
"begin\n"+
"␣␣␣␣shift_reg2␣<=␣{shift_reg2[98:0],␣led_template_signal2};\n"+
"end\n"+
"\n"
"assign␣counter1_max_value␣=␣\n")

for i in range(257, 285): #I'm only going to mess with the last two settings for now; come back later
    Output.append(text[i])
```

```
Output.append(
"(SW[3:0]==4'b1110)?\n"+
"    26'd"+ str(int((50000000/f1)-1))+":\n"+
"        26'd250;\n"+
"\n"+
"assign counter2_max_value = \n")
for i in range(257, 285): #I'm only going to mess with the last two settings for now; come back later
    Output.append(text[i])
Output.append(
"(SW[3:0]==4'b1110)?\n"+
"    26'd"+ str(int((50000000/f2)-1))+":\n"+
"        26'd250;\n")
firstnumber=50000000.0/(int((50000000/f1)))
secondnumber=50000000.0/(int((50000000/f2)))
Output.append("//The difference frequency for this run is "+str(np.abs(firstnumber-secondnumber))+"\n")

################################################################################

#We now have to actually provide the maximum value of the metacounter, which is
#just the number of cycles minus 1 (since 0 counts)

#For if you put in a single number for cycles_per_point:
metacounter1_max_value=[]
metacounter2_max_value=[]
for setting in range(4):
    if type(cycles_per_point1[setting]) is int:
            metacounter1_max_value.append( (len(coordinates1[setting])*cycles_per_point1[setting])-1)
#For if you put in an array:
    else:
            metacounter1_max_value.append(sum(cycles_per_point1[setting])-1)
    Output.append(
    "assign metacounter1_max_value = \n"+
    "7'd"+str((metacounter1_max_value[setting])) + ";\n")
    if type(cycles_per_point2[setting]) is int:
            metacounter2_max_value.append( (len(coordinates2[setting])*cycles_per_point2[setting])-1)
#For if you put in an array:
    else:
            metacounter2_max_value.append(sum(cycles_per_point2[setting])-1)
    Output.append(
    "assign metacounter2_max_value = \n"+
    "7'd"+str((metacounter2_max_value[setting])) + ";\n")
#more good code
for i in range (288, 309):
        Output.append(text[i])
################################################################################

Output.append(
"wire [63:0] output_bins1;\n"+
"assign output_bins1=\n")

Psi1=[]
Psi2=[]
for setting in range(4):
    Psi_setting1=[]
    Psi_setting2=[]
    for k in range(len(TYPES1[setting])):

            if TYPES1[setting][k]==0:
                    Psi_setting1.append(phaser.point(coordinates1[setting][k][0],
                    coordinates1[setting][k][1], z, f1))

            elif TYPES1[setting][k]==1:
                    Psi_setting1.append(phaser.line(coordinates1[setting][k][0],
                    coordinates1[setting][k][1], coordinates1[setting][k][2], z, f1)
                    )

            elif TYPES1[setting][k]==2:
                   Psi_setting1.append(phaser.dospuntos(coordinates1[setting][k][0]
                   , coordinates1[setting][k][1], coordinates1[setting][k][2],
                   coordinates1[setting][k][3], z, f1))

            elif TYPES1[setting][k]==3:
                   Psi_setting1.append(phaser.ring(coordinates1[setting][k][0],
                   coordinates1[setting][k][1], z, f1))

            elif TYPES1[setting][k]==4:
                   Psi_setting1.append(phaser.halfring(coordinates1[setting][k], 10
                   , z, f1))

            if TYPES2[setting][k]==0:
                    Psi_setting2.append(phaser.point(coordinates2[setting][k][0],
                    coordinates2[setting][k][1], z, f2))

            elif TYPES2[setting][k]==1:
                    Psi_setting2.append(phaser.line(coordinates2[setting][k][0],
                    coordinates2[setting][k][1], coordinates2[setting][k][2], z, f2)
                    )

            elif TYPES2[setting][k]==2:
                   Psi_setting2.append(phaser.dospuntos(coordinates2[setting][k][0]
                   , coordinates2[setting][k][1], coordinates2[setting][k][2],
                   coordinates2[setting][k][3], z, f2))

            elif TYPES2[setting][k]==3:
                   Psi_setting2.append(phaser.ring(coordinates2[setting][k][0],
                   coordinates2[setting][k][1], z, f2))

            elif TYPES2[setting][k]==4:
```

```
                Psi_setting2.append(phaser.halfring(coordinates2[setting][k],
                10, z, f2))
##############################################################################
    #In each case, we want to turn the resulting array of phases into an entry
    #in the array Psi. Then we have to put the phase configurations in a form
    #that the Verilog code can understand.
    Psi1.append(Psi_setting1)
    Psi2.append(Psi_setting2)
phases1 = []
phases2 = []
for k in range(len(coordinates1[setting])):
        Phases1= "{"

        for i in range(64):
                Phases1+="shift_reg1["
                Phases1+=str(Psi1[k][0][i])
                if i<63:
                        Phases1+="],"
                else:
                        Phases1+="]}\n"
        phases1.append(Phases1)

for k in range(len(coordinates2)):
        Phases2= "{"

        for i in range(64):
                Phases2+="shift_reg2["
                Phases2+=str(Psi2[k][0][i])
                if i<63:
                        Phases2+="],"
                else:
                        Phases2+="]}\n"
        phases2.append(Phases2)
##############################################################################

#now we just set the conditions under which we want each phase to be used, and
#give the code the list of phases.
#It really shouldn't matter, but I would like it if the focus moves to the
#points in the order they were input, which is why the  following method may
#look somewhat atypical.
metacounter1=metacounter1_max_value
metacounter2=metacounter2_max_value
#We're going to "simulate" a cycle of the metacounter, which starts at its
#maximum value and decreases every time the counter reaches zero.The first
#point should trigger as long as the metacounter's value is greater than its
#maximum value minus the number of cycles of the first step.

#Once again, we'll account for two cases: one in which the cycles_per_point is
#a number, and one in which it's an array:
##############################################################################
                        #Case 1: Number
for setting in range(4):
    if setting==0:
        switches="00"
    elif setting==1:
        switches="01"
    elif setting==2:
        switches="10"
    elif setting==3:
        switches="11"
    if type(cycles_per_point1[setting]) is int:
        for k in range(len(coordinates1[setting])):

            metacounter1_next = metacounter1[setting] - cycles_per_point1[setting]
            upper_limit = metacounter1[setting]+1
            if k+1 in range(len(coordinates1[setting])) :
                Output.append(
                "(SW[8:7] == 2'b"+switches+" && metacounter1<"+str(upper_limit)
                + " && metacounter1>" + str(metacounter1_next)+")?\n")
                Output.append("    "+str(phases1[k])+":\n")

            elif setting!=3:
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter1<"+
                str(upper_limit) + ")?\n")
                Output.append("    "+str(phases1[k])+":\n")
            else:
                Output.append("        "+str(phases1[k])+";\n")
            metacounter1[setting]=metacounter1_next

##############################################################################
                    #Case 2: Array

    else:
         for k in range(len(coordinates1[setting])):

            metacounter1_next = metacounter1[setting] - cycles_per_point1[setting][k]
            upper_limit = metacounter1[setting]+1
            if k+1 in range(len(coordinates1[setting])) :
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter1<"+
                str(upper_limit) + " && metacounter1>" + str(metacounter1_next)+
                ")?\n")
                Output.append("    "+str(phases1[k])+":\n")

            elif setting!=3:
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter1<"+
                str(upper_limit) + ")?\n")
                Output.append("    "+str(phases1[k])+":\n")
            else:
```

```
                Output.append("        "+str(phases1[k])+";\n")
            metacounter1[setting]=metacounter1_next
################################################################################
#Now to do the exact same thing for the second frequency:
                        #Case 1: Number

Output.append(
"wire [63:0] output_bins2;\n"+
"assign output_bins2=\n")
for setting in range(4):
    if setting==0:
        switches="00"
    elif setting==1:
        switches="01"
    elif setting==2:
        switches="10"
    elif setting==3:
        switches="11"
    if type(cycles_per_point2[setting]) is int:
        for k in range(len(coordinates2[setting])):

            metacounter2_next = metacounter2[setting] - cycles_per_point2[setting]
            upper_limit = metacounter2[setting]+1
            if k+1 in range(len(coordinates2[setting])) :
                Output.append(
                "(SW[8:7] == 2'b"+switches+" && metacounter2<"+str(upper_limit)
                + "&& metacounter2>" + str(metacounter2_next)+")?\n")
                Output.append("    "+str(phases2[k])+":\n")

            elif setting!=3:
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter2<"+
                str(upper_limit) + ")?\n")
                Output.append("    "+str(phases2[k])+":\n")
            else:
                Output.append("        "+str(phases2[k])+";\n")
            metacounter2[setting]=metacounter2_next

################################################################################
                    #Case 2: Array

    else:
         for k in range(len(coordinates2[setting])):

            metacounter2_next = metacounter2[setting] - cycles_per_point2[setting][k]
            upper_limit = metacounter2[setting]+1
            if k+1 in range(len(coordinates2[setting])) :
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter2<"+
                str(upper_limit) + "&& metacounter2>" + str(metacounter2_next)+
                ")?\n")
                Output.append("    "+str(phases2[k])+":\n")

            elif setting!=3:
                Output.append("(SW[8:7] == 2'b"+switches+" && metacounter2<"+
                str(upper_limit) + ")?\n")
                Output.append("    "+str(phases2[k])+":\n")
            else:
                Output.append("        "+str(phases2[k])+";\n")
            metacounter2[setting]=metacounter2_next
################################################################################
#Now we need to list off the LEDs
for i in range (64):
    #if ((i/8)>=4 and i%8>=4) or (i/8<4 and i%8<4):
    if (i/8)%2 == i%2:  #Checkerboard
    #if (i/16)%2==(i/2)%2: #4x4
    #if i/8>=5:#top-bottom split
    #if i%8>=4: #LR split

        net=1
    #elif i/8<=2:
    else: #no deadzone
        net=2
    Output.append(
    "assign{LED_reg["+str(i)+"]} = {1{output_bins"+str(net)+"["+str(i)+"]}};\n"
    )

#The rest of the code is good to go.
for i in range (490-356):
    Output.append(text[i+356])
################################################################################
#Now we just open the file that we're actually using to code the FPGA, and copy
#down the output.

g=open('phased_array_control.v', 'w')
for i in range(len(Output)):
    g.write(Output[i])
g.close()
################################################################################

#This completes the code. Hopefully this gives you the results you desire.

#Have a nice day!
```

C.2 Arduino Programs

C.2.1 automationcoderestart.ino

Written by Krutik Patel. This program allows you to specify a distance for the stepper motors to move, and is often used when aligning the system.

```
/*
This is the first revision of the arduino motor control script
that will be used to automate the photoacoustic measurements.
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AFMotor.h>

Adafruit_MotorShield AFMS = Adafruit_MotorShield();

Adafruit_StepperMotor *MotorX = AFMS.getStepper(200,1);
Adafruit_StepperMotor *MotorY = AFMS.getStepper(200,2);

int stepMadePin = 5 ;
int stepRevs = 10;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz

  MotorX->setSpeed(250);  // 15 rpm
  MotorY->setSpeed(250);
}

void loop() {
 /*
  Serial.println("Single coil steps");

  myMotor->step(200, FORWARD, SINGLE);
  myMotor->step(100, BACKWARD, SINGLE);
*/

  //y-back is up
  //x-back is towards you
  //MotorX->step(10*200,FORWARD,DOUBLE);
  //MotorX->step(40*200,FORWARD,DOUBLE);
  //MotorX->step(30*200, BACKWARD, DOUBLE);
  //MotorX->step(15*200, BACKWARD, DOUBLE);
  //MotorX->step(15*200, FORWARD, DOUBLE);
  //MotorX->step(30*200, FORWARD, DOUBLE);
 // MotorX->step(10*200, FORWARD, DOUBLE);
  //MotorY->step(60*200, BACKWARD,DOUBLE);
  MotorX->step(30*200, FORWARD    ,DOUBLE);
  //MotorX->step(30*200, BACKWARD,   DOUBLE);

  MotorX->release();
  MotorY->release();
  delay(1000000000);

}
```

C.2.2 automationlinescan.ino

Written by Krutik Patel. This program automates the process of collecting data along a single line, and is useful for alignment before performing a full data run.

```
/*
This is the second revision of the arduino motor control script
that will be used to automate the photoacoustic measurements.
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AFMotor.h>

//We begin with the set-up.

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *Motor = AFMS.getStepper(200,1);
//Adafruit_StepperMotor *MotorY = AFMS.getStepper(200,2);

int stepRevs = 2;

int totalSteps = 160;

int stepSize = 100;//step size 100 = half a rotation per step

int numSteps = totalSteps/stepRevs;

int lightsOnPin = 5;
int dataTakePin = 7;

int motorMoveWaitTime = 5000;
int dataWaitTime = 5000;
int coolDownTime = 2000;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz

  Motor->setSpeed(250);  // 50 rpm

  pinMode(lightsOnPin,OUTPUT);
  pinMode(dataTakePin,OUTPUT);
}

void takeData(){
  digitalWrite(lightsOnPin, HIGH);
  digitalWrite(dataTakePin, LOW);
  delay(motorMoveWaitTime);
  digitalWrite(dataTakePin, HIGH);
  delay(dataWaitTime);
  digitalWrite(dataTakePin,LOW);
  digitalWrite(lightsOnPin, LOW);
  delay(coolDownTime);
}



void loop() {
 /*
  Serial.println("Single coil steps");

  myMotor->step(200, FORWARD, SINGLE);
  myMotor->step(100, BACKWARD, SINGLE);
*/
  delay(10000);

  //First data taking, before anything else has happened.
  //We have to account for this edge case separately.

  Motor->step(stepRevs*stepSize*(numSteps/2), BACKWARD,DOUBLE);

  takeData();
  for (int j=0; j<(numSteps); j++){

    Motor->step(stepRevs*stepSize, FORWARD, DOUBLE);
    takeData();
  }


  //now bring us back to the front

  Motor->step(stepRevs*stepSize*(numSteps/2), BACKWARD,DOUBLE);
  Motor->release();


  delay(1000000000);
  delay(1000000000);
  delay(1000000000);

}
```

C.2.3 automationcodeforwardrev3.ino

This program is the most recent version of the code that, when uploaded to the Arduino, automates the data collection process by moving the stepper motors and controlling when the LEDs are on and when data is recorded. This program was more or less written by Krutik Patel, who wrote the previous version; updates to this version appear to have been planned but not implemented. On a related note, it should be mentioned that there is a bug which can cause the run to end prematurely if the number of rows in a data scan is even; this issue could be easily fixed, but was not.

```
/*
This is the second revision of the arduino motor control script
that will be used to automate the photoacoustic measurements.
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AFMotor.h>

//We begin with the set-up.

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *MotorX = AFMS.getStepper(200,1);
Adafruit_StepperMotor *MotorY = AFMS.getStepper(200,2);

int stepRevsX = 12;
int stepRevsY = 12;

int totalSteps = 360;

int stepSize = 100;

int numStepsX = totalSteps/stepRevsX;
int numStepsY = totalSteps/stepRevsY;

int lightsOnPin = 5;
int dataTakePin = 7;

int motorMoveWaitTime = 9000;
int dataWaitTime = 15000;
int coolDownTime = 5000;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz

  MotorX->setSpeed(250);  // 25 rpm
  MotorY->setSpeed(250);
  pinMode(lightsOnPin,OUTPUT);
  pinMode(dataTakePin,OUTPUT);
}

void takeData(){
  digitalWrite(lightsOnPin, HIGH);
  digitalWrite(dataTakePin, LOW);
  delay(motorMoveWaitTime);
  digitalWrite(dataTakePin, HIGH);
  delay(dataWaitTime);
  digitalWrite(dataTakePin,LOW);
  digitalWrite(lightsOnPin, LOW);
  delay(coolDownTime);
}



void loop() {
 /*
  Serial.println("Single coil steps");

  myMotor->step(200, FORWARD, SINGLE);
  myMotor->step(100, BACKWARD, SINGLE);
*/
  delay(10000);

  //First data taking, before anything else has happened.
  //We have to account for this edge case separately.

  MotorX->step(stepRevsX*stepSize*(numStepsX/2), BACKWARD,DOUBLE);
  MotorY->step(stepRevsY*stepSize*(numStepsY/2), BACKWARD,DOUBLE);

  takeData();
  for (int j=0; j<(numStepsY); j++){

    for (int i=0; i<numStepsX; i++){

      MotorX->step(stepRevsX*stepSize, FORWARD, DOUBLE);
      takeData();
    }
```

```
    MotorY->step(stepRevsY*stepSize ,FORWARD, DOUBLE);
    MotorX->step(stepRevsX*stepSize*(numStepsX/2), BACKWARD,DOUBLE);
        MotorX->step(stepRevsX*stepSize*(numStepsX/2), BACKWARD,DOUBLE);
        takeData();
  }

  for (int i=0; i<numStepsX; i++){
      MotorX->step(stepRevsX*stepSize, FORWARD, DOUBLE);
      takeData();
    }

  //now bring us back to the front

  MotorX->step(stepRevsX*stepSize*(numStepsX/2), BACKWARD,DOUBLE);
  MotorY->step(stepRevsY*stepSize*(numStepsY/2), BACKWARD,DOUBLE);


  MotorX->release();
  MotorY->release();


  delay(1000000000);
  delay(1000000000);
  delay(1000000000);

}
```

C.2.4 automationlettherebelight

The following code will cause the phased array to constantly output amplitude-modulated light.

```
/*
This is a revision of the second revision of the arduino motor control script
that will be used to automate the photoacoustic interference
measurements. This code will also tell the microphone when to trigger
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AFMotor.h>

//We begin with the set-up.

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *Motor = AFMS.getStepper(200,1);
//Adafruit_StepperMotor *MotorY = AFMS.getStepper(200,2);

int stepRevs = 2;

int totalSteps = 20;

int stepSize = 1;//step size 100 = half a rotation per step

int numSteps = totalSteps/stepRevs;

int lightsOnPin = 5;
int dataTakePin = 7;

int motorMoveWaitTime =1000;
int dataWaitTime = 36000;
int coolDownTime = 10000;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper␣test!");
  Serial.println("Off!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz

  Motor->setSpeed(250);  // 50 rpm

  pinMode(lightsOnPin,OUTPUT);
  pinMode(dataTakePin,OUTPUT);
}

void takeData(){
  digitalWrite(lightsOnPin, HIGH);
  digitalWrite(dataTakePin, LOW);
  delay(motorMoveWaitTime);
  digitalWrite(dataTakePin, HIGH);
  delay(1000);
  Serial.println("On!");
  delay(dataWaitTime-2000);
  Serial.println("Off!");
  delay(1000);
  digitalWrite(dataTakePin,LOW);
  digitalWrite(lightsOnPin, LOW);
  delay(coolDownTime);
}



void loop() {
 /*
  Serial.println("Single coil steps");

  myMotor->step(200, FORWARD, SINGLE);
  myMotor->step(100, BACKWARD, SINGLE);
*/
  delay(10000);

  //First data taking, before anything else has happened.
  //We have to account for this edge case separately.

  //Motor->step(stepRevs*stepSize*(numSteps/2), BACKWARD,DOUBLE);

  takeData();
  for (int j=0; j<(numSteps); j++){

    Motor->step(stepRevs*stepSize, FORWARD, DOUBLE);
    takeData();
  }


  //now bring us back to the front

  Motor->step(stepRevs*stepSize*(numSteps/2), BACKWARD,DOUBLE);
  Motor->release();


  delay(1000000000);
```

```
    delay(1000000000);
    delay(1000000000);

}
```

C.2.5 ManualScan.ino and MANUAL_MOVE.py

These two programs, when used together, will allow one to control the stepper motors manually through the computer connected to the Arduino. In order for this to work, one must first upload the Arduino code, ManualScan.ino, then run the Python program, MANUAL_MOVE.py, through the command prompt. One will then be able to control the stepper motors by entering numbers into the terminal; ManualScan.ino also lists what each number causes the stepper motors to do.

The following code, ManualScan.ino, when uploaded to the Arduino, will allow you to use a Python program to move the stepper motors manually.

```
/*
This code allows the user to perform manual scans of a fixed
step size in a given direction.  The lights will be left on
until the run is terminated.  This code should be run alongside
the python code MANUALCONT.py - although this code should be
uploaded to the Arduino first.
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <AFMotor.h>

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *MotorX = AFMS.getStepper(200, 1);
Adafruit_StepperMotor *MotorY = AFMS.getStepper(200, 2);

int stepRevs = 10;
int resetRevs=160;
int stepSize=100;
int lightsOnPin=5;
void setup() {
  Serial.begin(9600);
  AFMS.begin(); //default is 1.6 kHz, or give an argument in Hz
  MotorX->setSpeed(250);//50rpm
  MotorY->setSpeed(250);
  pinMode(lightsOnPin,OUTPUT);
  digitalWrite(lightsOnPin,HIGH);

}

void loop() {
 char command=Serial.read();
  if (command=='0'){
    digitalWrite(lightsOnPin,LOW);
  }
  if (command=='1'){
    MotorX->step(stepSize*stepRevs, FORWARD, DOUBLE);
  }
  if (command=='2'){
    MotorX->step(stepSize*stepRevs, BACKWARD,DOUBLE);
  }
  if (command=='3'){
    MotorY->step(stepSize*stepRevs, FORWARD,DOUBLE);
  }
  if (command=='4'){
    MotorY->step(stepSize*stepRevs, BACKWARD, DOUBLE);
  }
  if (command=='5'){
    MotorX->step(stepSize*resetRevs, FORWARD,DOUBLE);
  }
  if (command=='6'){
    MotorX->step(stepSize*resetRevs, BACKWARD,DOUBLE);
  }
  if (command=='7'){
    MotorY->step(stepSize*resetRevs, FORWARD,DOUBLE);
  }
  if (command=='8'){
    MotorY->step(stepSize*resetRevs, BACKWARD,DOUBLE);
  }
}
```

After uploading the above program to the Arduino, run the following Python program, MANUAL_MOVE.py from the command prompt, and you will be able to move the stepper motors by entering numbers into the command prompt.

```
import serial
arduino=serial.Serial('COM12', 9600, timeout=.1)
command=-1
while command!='0':
    command=raw_input("Input a number to issue a command.\n Numbers 1-4 correspond to steps, 5-8 to resets.  in order x+,
        ↪  x-, y+, y-.  0 to stop.\n")
    arduino.write(command)
```

C.3 Data Analysis Program

C.3.1 dataprocessingadditiveguilinescans.py

Written by Krutik Patel. This program analyzes data from the oscilloscope and produces a plot of the mean square pressure in arbitrary units as a function of position along a line scan. To use this program, one must create a file called "instruct.txt" which contains a few parameters about the line scan: how many files were produced, the number of data points, and the number of turns of the turn screw corresponding to the distance between data points.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 08 23:37:35 2014

@author: Krutik

This code generates a simple graph from a linescan of the photoacoustic set up.

The data files come from the MATH channel on the o-scope, and the algorithm for
creating the plot is the same as in the main image code, except its simpler because
its only a line and doesn't have to worry about the path.
"""

#get libraries
import matplotlib.pyplot as plt
import numpy as np
import Tkinter

from tkFileDialog import *


#Program is written here. It's done within a function to make it callable from the GUI

def processData():

    #load instruct.txt, which contains information about the run.

    instruct = np.genfromtxt(datadirectory + "/instruct.txt", unpack=True,delimiter=',')

    numcsv =  int(instruct[0]) #number of csv files
    totalturns= int(instruct[1]) #total length of run in turns
    stepsize = float(instruct[2]) # step size in turns

    numsteps = (totalturns/stepsize)  #number of steps
    numpoints = numsteps + 1 #number of points
    threshold = -3 #threshold to distinguish between HIGH and LOW average. Tweak for data.
    maxmeas=10 #maximum possible number of measurements from o-scope per point. This is roughly
               #data take time / 3 sec, but its not always reliable. Overestimating is fine.

    cmperturn = .0254 #100TPI actuator screw

    axeslength = totalturns * cmperturn #length of axes on image



    #create empty arrays for data storage and enumeration
    datalist = []
    datalistavg=[]
    datamatrix = np.zeros([numpoints,maxmeas])

    def AVG(list): #take an average
        return np.sum(list)/len(list)

    def RMS(list): #take an RMS
        return np.sqrt(np.sum(list**2)/len(list))

    for i in range(1,numcsv+1):
        #Load in data
        #datai = np.genfromtxt( datadirectory + "/data (" + str(i) +").csv", unpack=True,delimiter=',',skip_header=18,
              ↪ usecols=(3,4))
        datai=np.genfromtxt(datadirectory+"/data (" + str(i) + ").csv", unpack=True, delimiter=',', usecols=(3, 4))
        if i==1:
            print datai
        datairms = RMS(datai[1]) #average it
        dataiavg = AVG(datai[1]) #rms it
        datalist.append(datairms**2 - dataiavg**2) #this is the rms of the data itself, without the offset.
        datalistavg.append(dataiavg) #store it
        print(i) #just to keep track of progress

    datalist=np.array(datalist) #turn into np array for speed
    datalistavg=np.array(datalistavg)

    j=0 #index for position
    m=0 #index for measurement number

    # if the raw data's average is above the threshold, i.e. real data being taken, then keep it
    for i in range(0,len(datalist)):
        if datalistavg[i]>threshold and m<maxmeas:
            datamatrix[j,m]=datalist[i]
            m=m+1
```

```
    #if the state of it has changed, increment position
        if datalistavg[i]<threshold and datamatrix[j,0]>0:
            if j<numsteps:
                j = j+1
            m=0

    #make actual curve
    ydata = np.zeros([numpoints])
    avgTemp = []

    #trim off the zeroes and average measurements
    for i in np.arange(0,numpoints):
        for z in np.arange(0,maxmeas):
            if datamatrix[i,z] != 0:
                avgTemp.append(datamatrix[i,z])
        q=max(1, len(avgTemp))
        avg = sum(avgTemp)/q
        avgTemp=[]
        ydata[i]=avg
        avg=0



    #np.savetxt(datadirectory+"/plot.csv", plot, delimiter=",") #if it is desirable to save it

    #plot a line and the points
    plt.plot(np.arange(0,numpoints),ydata)
    plt.plot(np.arange(0,numpoints),ydata,'bo')
    plt.show()
    print ydata[0]



#GUI stuff. Not important, just creates a basic interface to specify a
#directory and hit a button that runs the program. Just convenience.

top = Tkinter.Tk()
top.wm_title('Generate␣Sound␣Field␣Image')



datadirectoryL=Tkinter.Label(top,text='Choose␣Data␣Folder')
datadirectoryL.pack()
datadirectory=askdirectory(parent=top)

startButton = Tkinter.Button(top, text='Make␣Image', command =
                             processData)
startButton.pack()



top.mainloop()
```

C.3.2 dataprocessingadditivegui.py

Written by Krutik Patel. This program functions similarly to the previous program, but analyzes data from a full data run rather than a line scan and plots the data accordingly.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 08 23:37:35 2014

@author: Krutik

This code is intended to take the data harvested by the 3-sec interval
data logging capability on the o-scope and produce a usable intensity
field to plot.

The general idea is that the o-scope will output the MATH channel, which
will be the sum of the output of digital pin 5 on the Arduino and
the actual signal. This python script will look at both the RMS and the average
of the data, which will allow it assess the DC offset given by pin 5. So, by
using this pin to indicate when it has stepped, and by hardcoding the
simple snaking path the sensor takes, an image of the field can be plotted.

"""


#get libraries
import matplotlib.pyplot as plt
import numpy as np
import Tkinter
from tkFileDialog import *


#Program is written here. It's done within a function to make it callable from the GUI


def processData():

    #load information about the run from instruct.txt
    instruct = np.genfromtxt(datadirectory + "/instruct.txt", unpack=True,delimiter=',')

    numcsv =  int(instruct[0]) #number of files
    totalturns= int(instruct[1]) # total number of turns
    stepsizex = float(instruct[2]) #step sizes in turns
    stepsizey = float(instruct[3])
    startpointx = float(instruct[4]) #start points. Not always used, sets axes correctly.
    startpointy = float(instruct[5])
    numstepsx = (totalturns/stepsizex) #number of steps
    numstepsy = (totalturns/stepsizey)
    numpointsx = numstepsx + 1 #number of points
    numpointsy = numstepsy + 1
    threshold = 0.01 #this is going to have to be tweaked depending on the data, PAY ATTENTION TO THIS LINE
        ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!
    maxmeas=10 #total number of possible measurements. Roughly wait time/ 3 sec but unreliable.
            #Overestimate.

    cmperturn = .0254     #100TPI actuator screw

    axeslength = totalturns * cmperturn #total axes length

    datamatrix = np.zeros([numpointsx,numpointsy,maxmeas]) #create empty data array

    datalist = [] #empty lists for storage of data
    datalistavg=[]

    def AVG(list): #Average function
        return np.sum(list)/len(list)

    def RMS(list): # RMS Function
        return np.sqrt(np.sum(list**2)/len(list))

    for i in range(1,numcsv+1):
        #load the data, and save both the average and the RMS. The arduino changes the
        #average whenever a step has been made, so using this information and knowledge of the path,
        #the image is recreated.

        datai = np.genfromtxt( datadirectory + "/data_(" + str(i) +").csv", unpack=True,delimiter=',',skip_header=18,
            ↪ usecols= (3,4))
        datairms = RMS(datai[1])
        dataiavg = AVG(datai[1])
        datalist.append(datairms**2 - dataiavg**2) #signal RMS squared ~ intensity
        datalistavg.append(dataiavg)
        print(i) #for keeping track

    datalist=np.array(datalist) #turn into np arrays for speed
    datalistavg=np.array(datalistavg)

    j=numstepsx # x index
    k=numstepsy # y index
    m=0 #measurement index
    right = False #direction index

    #this set of loops moves along the path in the correct order
    for i in range(0,len(datalist)):
        if k== -1:
            break
        if datalistavg[i]>threshold and m<maxmeas:
```

```
            datamatrix[j,k,m]=datalist[i]
            m=m+1

        #shift to next collection, zig-zagging along the data taking path
        if datalistavg[i]<threshold and datamatrix[j,k,0]>0:
            if right and j < numstepsx :
                j=j+1
            elif right and j == numstepsx :
                right = False
                k = k-1
            elif not right and j > 0:
                j = j - 1
            elif not right and j==0:
                right = True
                k = k-1
            m=0


    #make empty image array
    image = np.zeros([numpointsx,numpointsy])
    avgTemp=[]

    #trim off zeroes and take averges of measurements
    for x in np.arange(0,numpointsx):
        for y in np.arange(0,numpointsy):
            for z in np.arange(0,maxmeas):
                if datamatrix[x,y,z] != 0:
                    avgTemp.append(datamatrix[x,y,z])
            #TEMPORARY FIX INCOMING
            q=max(len(avgTemp), 1)
            avg = sum(avgTemp)/q
            #END TEMPORARY FIX
            #print(str(i)+"-"+str(avg))
            avgTemp=[]
            image[x,y]=avg
            avg=0



    np.savetxt(datadirectory+"/image.csv", image, delimiter=",") #save image
    #Plotting

    plt.subplots()
    plt.subplot(121)

    plt.imshow(np.flipud(np.transpose(image)),cmap='hot',extent=[startpointx - (axeslength/2)
                            ,startpointx + (axeslength/2) ,startpointy - (axeslength/2),
                             startpointy + (axeslength/2)],interpolation='none') #plot
    plt.title('Intensity (a.u.)')
    plt.xlabel('X-Position (cm)')
    plt.ylabel('Z-Position (cm)')


    plt.colorbar()

    plt.subplot(122)
    plt.title('Raw Data')
    plt.xlabel('Iteration')
    plt.ylabel('Voltage')
    plt.plot(np.arange(1,len(datalist)+1),datalist,'ro',label='RMS data')
    plt.plot(np.arange(1,len(datalistavg)+1),datalistavg,'bo', label='Avg data')
    plt.legend()

    plt.show()

#No important code here, just set up for the GUI to make it convenient to navigate to the
#data directory.
top = Tkinter.Tk()
top.wm_title('Generate Sound Field Image')



datadirectoryL=Tkinter.Label(top,text='Choose Data Folder (image!)')
datadirectoryL.pack()
datadirectory=askdirectory(parent=top)

startButton = Tkinter.Button(top, text='Make Image', command =
                             processData)
startButton.pack()



top.mainloop()
```

C.4 Miscellaneous

C.4.1 EfficiencyCalc.py

This program performs the efficiency calculation discussed in the last section of Chapter 3. This program also performs the calculation by an alternate method, basing the amount of power consumed by the LEDs on specifications in the data sheet rather than direct measurement. The approximations used for the luminosity funciton and the spectral energy density are defined, and a plot of the latter is produced for comparison to the information provided on the datasheet.

```
"""

Created January 26, 2016

@author: Morgan Brubaker

The purpose of this code is to take an image file of 40 kHz ultrasound
generated by the dataprocessingadditivegui code, determine the acoustic energy
flux through the image, and compare to the optical energy output.  This estimate
is dependant on the calculation of the conversion from squared v_rms to pressure
, which is why, for now, this calculation only holds for 40 kHz ultrasound.
This code may borrow heavily from the dataprocessingadditivegui code in how the
data is processed"""
#get libraries
import matplotlib.pyplot as plt
import numpy as np
import Tkinter
from tkFileDialog import *



#like the data processing file, we define a function that is callable from the
#GUI
def EfficiencyCalc():
    #We also need some of the info from instruct.txt:
    instruct = np.genfromtxt(datadirectory + "/instruct.txt", unpack=True,delimiter=',')
    totalturns= int(instruct[1])
    stepsizex = int(instruct[2]) #step sizes in turns--I will assume is equal to stepsizey
    numstepsx=(totalturns/stepsizex)
    numpointsx = numstepsx + 1
    mperturn = .000254     #100TPI actuator screw
    Spacing = stepsizex*mperturn

    #First, we define the process by which we calculate the power represented by
    #each data point.
    def vvtopower(a, DeltaX):
        p = np.sqrt(a)/20.0 #Convert data to rms pressure
        p = p**2 #square rms pressure
        p = p*(DeltaX)**2 #multiply by the area of each "pixel"
        p = p/1.225 #divide by air density
        p = p/343.59 #divide by speed of sound
        return p

    #Next, we calculate the LED power:


    n = 500000  #feel free to change n when testing this calculation; increasing n
                #should make the calculation of optical power more accurate but slower

    #We start by writing the functions we use to approximate y(lambda), J(lambda), and their product.
    #We also plot y and J so that we can compare the approximation used to the actual functions
    c1=139.7/(2*np.sqrt(2*np.log(2)))
    c2=27/(2*np.sqrt(2*np.log(2)))
    x1=np.linspace(400, 700, n+1)
    x2=np.linspace(380, 830, 1.5*n + 1)
    #y1 is the product of the luminosity function y and the relative spectral power distribution J
    y1=(np.e**(-(x1-555)**2/(2*(42.4661)**2)))*(np.e**(-(x1-605)**2/(2*(c1)**2)) + (385.0/600.0)*np.e**(-(x1-455)**2/(2*(
        ↪ c2)**2)))
    #y2 is the relative spectral power distribution J
    y2=(np.e**(-(x2-605)**2/(2*(c1)**2)) + 0.64*np.e**(-(x2-455)**2/(2*(c2)**2)))
    #y3 is the luminosity function y
    y3=(np.e**(-(x1-555)**2/(2*(42.4661)**2)))

    plt.plot(x2, y2, color='r')#J is plotted in red
    plt.plot(x1, y3, color='g')#y is plotted in green
    plt.show()

    #Now to start the actual calculation:

    #First we calculate lumens using measured numbers and electrical efficiency
    Vres=0.920
    R=2.2
    I=Vres/R
    Vled=0.80
    P_el=I*Vled
    #The above calculation gives the electrical power dissipated across the LED.
    #According to sources online, one can determine the electrical efficiency of
    #an LED in lumens/watt by dividing the luminous output given on the datasheet
    #by the product of the maximum voltage current drop across the LED and the
    #test current.  The corresponding numbers on the datasheet are 87.4 lumens,
    #3.9V, and 350mA.  Therefore,
```

```
    El_Ef=87.4/(3.9*0.350)
    lumens1=P_el*El_Ef
    #We also calculate lumens from the datasheet using our current
    lumens2=1.15*(87.4)
    #Then we calculate the luminous efficacy which is 683.002 lm/W times the ratio of the integrals
    #of y(lambda)J(lambda) and J(lambda), where y is the luminosity function and J is the spectral
    #power distribution.
    YJintegral = 0
    Jintegral = 0
    for y in y1:
        YJintegral += (y*300/n) #note that for both integrals, 300/n is the
    YJintegral-=(y1[n])*300/n #subtracting off right end term; effectively doing a left Riemann sum
    print ("YJintegral = "+str(YJintegral))
    for y in y2:
        Jintegral += (y*300/n)
    Jintegral -=(y2[1.5*n])*300/n#subtracing off end term
    print ("Jintegral = "+str(Jintegral))
    LuminousEfficacy=683.002*(YJintegral/Jintegral)
    Watts1=lumens1/LuminousEfficacy
    Watts2=lumens2/LuminousEfficacy
    print ("Watts1="+str(Watts1))
    print ("Watts2="+str(Watts2))
    Watts1=.13*Watts1#because 74% of light passed through film for the image in question, and the LEDs are off half the
        ↪ time
    Watts2=.13*Watts2

    #now, we calculate the sound power
    totalPower=0
    MaxPressure=0
    data=np.genfromtxt( datadirectory + "/image.csv", unpack=True,delimiter=',')
    for i in range(numpointsx):
        for j in range(numpointsx):
            q=data[i,j]
            thispower=vvtopower(q, Spacing)
            totalPower+=thispower
            #we also find the maximum pressure for the image
            p=np.sqrt(q)/20.0
            if p>MaxPressure:
                MaxPressure=p
    print ("The maximum pressure is "+str(MaxPressure))
    print ("The total power is "+str(totalPower)+" W")
    print ("The spacing is "+str(Spacing)+" m")
    Eff1=totalPower/Watts1
    Eff2=totalPower/Watts2
    print("Eff1="+str(Eff1)+" and Eff2="+str(Eff2)+".")

#The rest of this is copy-pasted from the analysis code:
#No important code here, just set up for the GUI to make it convenient to navigate to the
#data directory.
top = Tkinter.Tk()


datadirectoryL=Tkinter.Label(top,text='Choose Data Folder (image!)')
datadirectoryL.pack()
datadirectory=askdirectory(parent=top)

startButton = Tkinter.Button(top, text='Make Image', command =
                             EfficiencyCalc)
startButton.pack()



top.mainloop()
```

Appendix D

Publication

The remainder of this document contains a manuscript of the publication in Applied Physics Letters regarding the work discussed in this thesis.

Photoacoustic ultrasound sources from diffusion-limited aggregates

Krutik Patel,[1] Morgan Brubaker,[1] Alexander Kotlerman,[1] Robert Salazar,[1] Eli Wolf,[1] and David M. Weld[1, a)]
Department of Physics, University of California, Santa Barbara, California 93106, USA

(Dated: 11 October 2016)

Metallic diffusion-limited aggregate (DLA) films are well-known to exhibit near-perfect broadband optical absorption. We demonstrate that such films also manifest a substantial and relatively material-independent photoacoustic response, as a consequence of their random nanostructure. We theoretically and experimentally analyze photoacoustic phenomena in DLA films, and show that they can be used to create broadband air-coupled acoustic sources. These sources are inexpensive and simple to fabricate, and work into the ultrasonic regime. We illustrate the device possibilities by building and testing an optically-addressed acoustic phased array capable of producing virtually arbitrary acoustic intensity patterns in air.

PACS numbers: 8.20.Pa, 68.35.Iv, 05.45.Df, 43.35.+d
Keywords: Photoacoustic, ultrasound, nanostructures

I. INTRODUCTION

Stochastically nanostructured materials exhibit optical, electronic, and thermal properties which can differ drastically from those of ordered nanostructures and of homogeneous samples of the same material. In nanostructured aggregates, a combination of strong broadband optical absorption and fast local thermal response gives rise to a strong photoacoustic effect, in which amplitude-modulated light is transduced to sound at the frequency of amplitude modulation. In this work we theoretically and experimentally investigate the photoacoustic response of nanostructured aggregates in air, and demonstrate that such materials enable the construction of acoustic sources including an optically-addressed photoacoustic phased array. The flexibility of such acoustic sources could enable low-cost generation of arbitrary broadband acoustic and ultrasonic waveforms, with potential applications ranging from subdiffraction air-coupled acoustic sources to materials characterization.

The study of photoacoustic phenomena in bulk materials has an illustrious history[1], and the effect has proven useful as a characterization tool in a variety of contexts[2]. Photoacoustic material response has also been explored as an optically-controlled source of sound waves: for example, photoacoustic sources have been fabricated from lithographically-defined nanostructures, thin films, multilayer polymer-metal composites, nanoparticle-filled epoxies, and carbon nanotubes[3–7]. In this work, we discuss a simple and low-cost technology for air-coupled photoacoustic ultrasound generation based on DLA films. The technology is within the capabilities of nearly any laboratory, requiring only very moderate (few torr) vacuum conditions, inexpensive metal sources, and simple modulated LEDs. We illustrate potential applications by constructing an optically-addressed photoacoustic phased array capable of producing patterned ultrasonic fields at far lower complexity and cost than a conventional multiple-piezoelectric-source phased array.

a)Corresponding author; Electronic mail: weld@physics.ucsb.edu

II. SAMPLE FABRICATION & MORPHOLOGY

Highly disordered nanostructured films of many evaporable substances can be simply and reproducibly prepared by thermal evaporation in the presence of a low-pressure background of inert gas. In films prepared using this procedure, a complex granular structure results from the randomization of atomic velocities before aggregation in the gas phase[8–12]. Such films often but not always resemble the product of diffusion-limited aggregation. Diffusion-limited aggregation occurs when particles executing a random walk collide and stick together; this process occurs in a wide variety of physical contexts, and is expected to produce a treelike fractal structure[13]. As a concrete example, we find that nanostructured bismuth films are reliably produced in a standard thermal evaporator by running 115 A of current through a 9 mΩ alumina-coated molybdenum evaporation boat for 4 minutes, in a background pressure of 2 Torr of Argon. The nanostructured films are deposited on any surface above the boat, although for surfaces too close to the boat they tend to melt and coarsen. The properties of the film do not depend sensitively on deposition parameters. Material costs for such a deposition are on the order of a few cents for a common evaporant like bismuth, and the total time to produce a film is less than ten minutes once the system is pumped down.

Although much of the research into such films has used gold as the evaporant[10–12], we find that DLA films are easily prepared from virtually any evaporable substance. In a sense, this method achieves its generality and relative material-independence by using classical mechanics rather than chemistry to produce structures at the nanoscale. We have used this technique to prepare DLA films of copper, silver, aluminum, iron, gold, indium, nickel, bismuth, tin, germanium, and silicon monoxide.

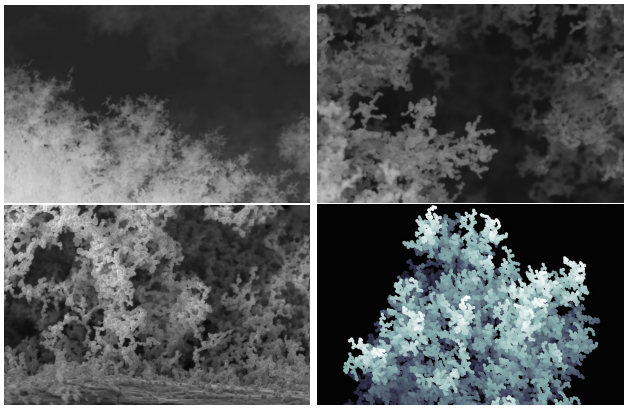

FIG. 1. Morphology of random nanostructured films. Electron micrographs of (clockwise from lower left) stochastically nanostructured bismuth, aluminum, and copper films, and results of numerical simulation of diffusion-limited aggregation. All SEM magnifications are the same, and each image is approximately 1.8 μm high.

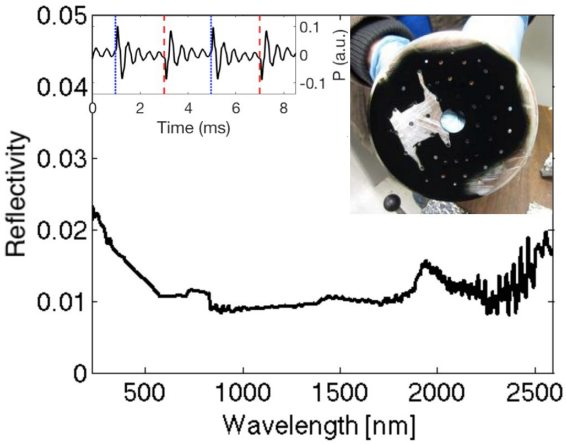

FIG. 2. Reflectivity versus wavelength of nanostructured copper film. Right inset: Photograph of the measured film on a sample holder. The bare area was covered by a sample during deposition. Left inset: Sound pressure versus time for a film exposed to a 250 Hz square-wave optical amplitude modulation. When the light turns on (off) at dotted (dashed) lines, the sound pressure rises (drops) rapidly. The subsequent oscillations are a consequence of ringing in the microphone.

Films prepared using copper and bismuth appear to be stable over several years when stored in air at room temperature. Elevated temperatures or very high optical intensities (at least two orders of magnitude higher than those used to obtain the results presented below) can rapidly decrease the optical absorptivity of the films, presumably via oxidation or coarsening of the nanostructure. At somewhat lower optical intensities of roughly 0.1-1 kW/m^2 bismuth films display photo-induced aging over the course of weeks, during which the absorptivity and photoacoustic efficiency are gradually reduced, though not to zero. We expect that this behavior is strongly material-dependent.

Fig. 1 compares electron micrographs of samples produced with this technique to numerically-generated diffusion-limited aggregates. The qualitative agreement between the observed structure and the predicted morphology of diffusion-limited aggregates is quite good for materials like aluminum, copper, and bismuth. The most striking visual aspect of films prepared in this way is their extreme blackness: as shown in Fig. 2, they typically absorb 99% of incident light across a broad range of wavelengths[14]. If the light is amplitude-modulated, the films also exhibit a strong photoacoustic response. Although the photoacoustic phenomena modeled and presented here all occur in air, fluid-coupled ultrasound is a possible future direction of study. For this reason we note that DLA films of copper and bismuth, though they are somewhat fragile and (in the case of copper) extremely hydrophobic, can easily be immersed in fluids without damage. In the next section, we discuss a simple model of the air-coupled photoacoustic properties of a nanostructured granular material.

III. PHOTOACOUSTIC PROPERTIES: THEORY

Our qualitative model of the air-coupled photoacoustic response of DLA films assumes that the films absorb nearly all light incident upon them, converting the energy into heat. We consider the effect of this absorption on a single grain of material at the smallest (material-dependent) length scale which characterizes the structure. Because of the diffuse fractal nature of the material, individual grains are very weakly thermally coupled to the rest of the film. This means that the grains, which have a small heat capacity due to their small size, can change their temperature substantially and rapidly in response to sudden illumination. When the illumination stops, our model assumes that the grains lose heat by conduction into the surrounding gas. This is in contrast to a three-dimensional bulk metal, in which conductive losses into the rest of the material would generally be more important. Amplitude-modulated light can thus excite a large-amplitude oscillation in the temperature of the gas near individual grains at the surface, which leads to a pressure oscillation and the emission of sound waves into the gas. This process should occur for essentially any material, and in the 1 KHz range we have observed audible responses from other black materials such as carbon soot. However, the efficiency of photoacoustic generation by this mechanism will fall off rapidly above some maximum frequency which depends on thermal time constants of the specific nanostructure. For DLA films, both this maximum frequency and the expected amplitude of temperature changes are large, leading to a strong photoacoustic response in air in the ultrasound regime.

To quantitatively elucidate this simple model, we consider how quickly an individual grain in a DLA film can thermally equilibrate with its surroundings, assuming that the dominant thermal relaxation mechanism is conductivity into the neighboring gas. The rate of thermal energy loss from a grain of metal due to conduction into the neighboring gas is

$$\frac{dE}{dt} = C\frac{dT}{dt} = \alpha\ \Gamma\ k_{\mathrm{B}}\Delta T, \qquad (1)$$

where C is the heat capacity of the grain, $\alpha \simeq 0.15$ is a numerical factor parametrizing the degree of thermalization of a gas atom after a single collision with a surface, Γ is the collision rate of gas atoms with the surface, and ΔT is the temperature difference between the grain and the gas. The collision rate $\Gamma = nva$ is the number density of the gas n times the RMS velocity v of a gas atom times the area a of the grain, which for a 10-nm-radius grain at STP is about 4 THz. The heat capacity C depends on the substance and grain size; for a 10-nm bismuth grain it is around $5{\times}10^{-18}$J/°K. There is therefore a material-dependent upper bound on the frequency at which such grains can change their temperature, given by $\frac{1}{\tau} = \frac{\alpha\Gamma k_B}{C}$. This result suggests that the bismuth grain can change its temperature at rates greater than 1 MHz.

This model suggests directions for further optimization. A 1nm grain would respond 10 times faster than a 10nm grain (100× less area, 1000× less heat capacity). As shown in Fig. 1, the grain size is material-dependent; aluminum, for example, produces much smaller grains than bismuth under similar deposition conditions. Operating in helium instead of air would decrease τ by a factor of 3 due to the higher mean velocity, and operating at higher pressure would decrease τ linearly.

At frequencies below $1/\tau$, the amplitude of the photoacoustic response depends upon the amplitude of the light-induced temperature oscillation. For light with intensity I and a grain with surface area A_{grain}, the rate of change of temperature is

$$\frac{d\Delta T}{dt} = I \cdot A_{\mathrm{grain}} - \frac{\alpha\Gamma k_B}{C}\Delta T. \qquad (2)$$

In the steady state the optical heating rate balances the cooling rate. For square-wave light modulation at frequencies far below $1/\tau$, we observe that the films respond with a sharp positive and negative pressure pulse at the light switching times, as shown in the inset of Fig. 2. This is consistent both with our simple model and thorough theoretical analysis[15]. Possible resonant effects[16] are an intriguing area for future study.

This simple model ignores effects which are certainly present at some level, such as radiative loss or conductive loss into the material[2]. The model does not predict the fraction of optical energy converted to acoustic energy, which we experimentally estimate to be on the order of 10^{-8}. For reference, this is about 3.5 orders of magnitude greater than the photoacoustic efficiency reported for a metal-coated silicon wafer in air[17]. Higher efficiencies are attainable in water, although our efficiency is comparable to that reported for metallic films[5] and nanotube-based absorbers[3] in water. Investigation of the relative frequency-dependent efficiencies of different photoacoustic techniques is a promising direction for further research.

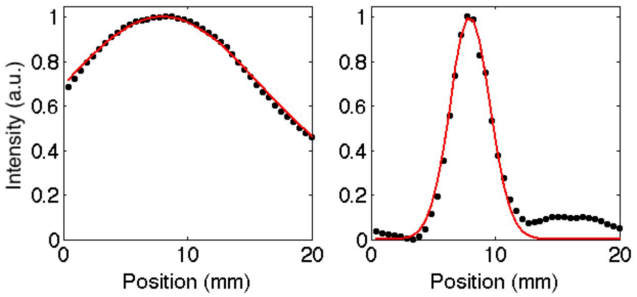

FIG. 3. Diffraction-limited focused ultrasound from a DLA film. Nanostructured bismuth-black approximately 3 μm thick was deposited on a plano-concave lens with a 150-mm radius of curvature and back-illuminated with amplitude-modulated white LEDs at an optical intensity of around 140 W/m^2. Acoustic intensity was measured with narrowband ultrasonic transducers: TCT40-16R for 40 kHz and SensComp 200KHF18 for 200 kHz, both using a PAR 113 preamplifier. Points show measured acoustic intensity versus position. Lines are gaussian curves with a frequency-dependent diffraction-limited width $w_0(\nu)$, scaled only vertically and centered on the observed spot. Left panel is a 40KHz focus, and right panel is a 200 KHz focus. These data were obtained by summing 2D acoustic intensity scans in one direction.

IV. PHOTOACOUSTIC PROPERTIES: EXPERIMENT

The photoacoustic response of all DLA films we have produced is strong enough that illumination of a flat film with an ordinary LED flashlight which uses amplitude modulation in the acoustic range produces a response audible to the human ear. More quantitatively, the strength of the photoacoustic response at 40 KHz is more than an order of magnitude stronger than that of a black graphite surface.

The simplicity of DLA-based photoacoustic sources enables the construction of flexible and powerful acoustic devices. The simplest incarnation of a directional DLA-based photoacoustic source is a plano-concave spherical lens with a nanostructured film deposited on the concave side. In such a device, the acoustic wavefronts generated by the DLA film inherit the curvature of the lens surface, leading to a converging acoustic beam[3]. As a proof of concept, we deposited nanostructured bismuth-black on a plano-concave lens and back-illuminated it with amplitude-modulated light to create a focused acoustic source. Results for modulation frequencies of 40 KHz and 200 KHz (chosen based on available transducers) are pre-

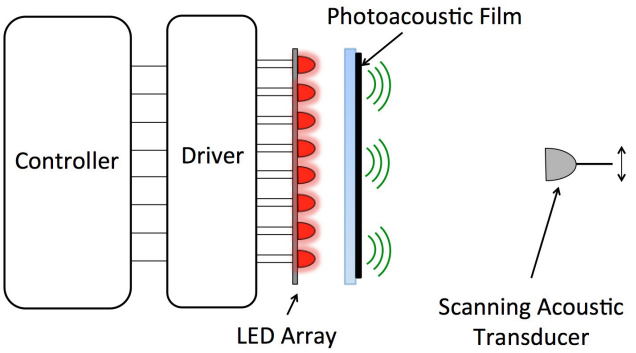

FIG. 4. Diagram of photoacoustic phased array apparatus. The field-programmable gate array controller creates 64 amplitude-modulated voltage signals with varying phase. These are converted to current signals by a FET driver and sent to the 8-by-8 LED array (100 mm on a side) for conversion to intensity-modulated light. The photoacoustic film converts the amplitude-modulated light into sound with a controllable position-dependent acoustic phase. Propagation of the resulting wavefront creates an intensity pattern which depends on the chosen phase pattern. After some propagation distance, the acoustic intensity as a function of position is detected by a scanning acoustic transducer.

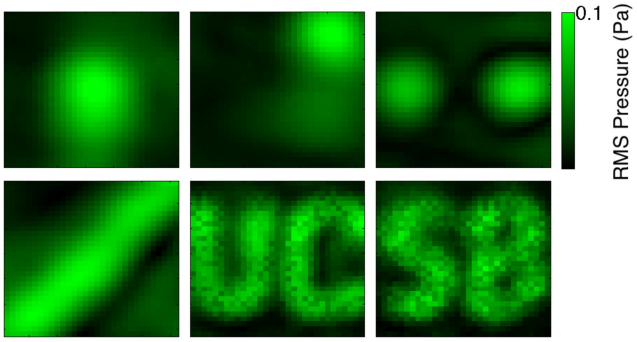

FIG. 5. Patterned acoustic pressure fields created by the optically addressed phased array, as measured by a scanning piezo transducer. Images demonstrate the ability to create moving foci, multiple foci, and virtually arbitrary intensity patterns. To create these images, 140 W/m^2 of white light was used to back-illuminate bismuth-black films with a thickness of about 3 μm. The acoustic frequency is 40 KHz, and all images are 30 mm across except the "UCSB" images, which are 45mm across. All images were taken at a distance of 70mm from the DLA film. Colorbar indicates the measured pressure for the upper-right figure. This pressure (0.1 Pa RMS) is representative of the typical range, though colormaps on the other subplots vary for visibility.

sented in Fig. 3, and compared to calculated diffraction-limited intensity curves with $1/e^2$ radius $w_0 = 2cF/\pi\nu D$, where c is the speed of sound, F is the focal length, ν is the frequency, and D is the lens diameter. The foci are found to be diffraction-limited; as expected, the 200 KHz spot is 5 times smaller than the 40KHz one.

To further explore the flexibility of acoustic sources based on DLA films, we constructed an optically-addressed acoustic phased array, diagrammed in Fig. 4. A field-programmable gate array in combination with an array of FET-based current switches is used to individually control the phase of amplitude modulation for each high-power LED in an 8-by-8 square array with a pitch of 12mm. The array is mounted directly behind a glass panel covered with nanostructured bismuth. This configuration allows for virtually arbitrary optical control of the acoustic wavefronts generated at the DLA film. Like the recently-proposed metascreen-based passive phased array[18], this device avoids the complexity and cost of a multiple-acoustic-source phased array, but in an entirely different manner. Here the technical advantages arise from the comparative ease and flexibility of generating amplitude modulated light, as compared to direct generation of ultrasound.

Figure 5 shows a few intensity patterns created by the photoacoustic phased array and measured with a scanning acoustic transducer. With an appropriately chosen phase configuration, the device can mimic the photoacoustic lens, creating a single focused acoustic intensity maximum as shown in the upper-left panel of the figure. By adding a phase offset which depends linearly on position, this virtual lens can be dynamically tilted, allowing for full spatiotemporal control of the acoustic intensity maximum. Such a translated focus is shown in the upper-middle panel of Fig. 5. Slightly more complex phase configurations can create multiple foci (upper-right panel of Fig. 5) or lines of maximum acoustic intensity (lower-left panel of Fig. 5). Arbitrary spatial variation of the phase pattern can be used to create more complex waveforms, with a precision limited by acoustic diffraction and the spatial period of the LED array. The fact that the 12mm pitch is larger than half a wavelength of 40 KHz ultrasound also limits the photoacoustic efficiency due to radiation of power into side lobes. Time-varying phase configurations allow for "painting" of time-averaged acoustic intensity patterns, for example by scanning a focused spot. The "UCSB" images in Fig. 5 were generated in this way, at a repetition rate greater than 200 Hz.

V. CONCLUSION

We have discussed, modeled, and demonstrated air-coupled photoacoustic sources based on nanostructured diffusion-limited aggregates. These sources are inexpensive and simple to fabricate and are capable of broadband ultrasound generation. As an illustration of the device possibilities, we designed and constructed a DLA-based optically-addressed ultrasonic phased array and used it to create virtually arbitrary acoustic intensity patterns.

ACKNOWLEDGMENTS

The authors thank Thomas Witten for a useful and interesting conversation; Carl Felten, Daeseong Kim, and Andrew Williams for experimental assistance in the construction of the acoustic phased array driver; and the Hellman Family Faculty Fellowship for support.

[1] A. G. Bell, American Journal of Sciences **XX**, 305 (1880).
[2] H. Vargas and L. Miranda, Physics Reports **161**, 43 (1988).
[3] H. W. Baac, J. G. Ok, A. Maxwell, K.-T. Lee, Y.-C. Chen, A. J. Hart, Z. Xu, E. Yoon, and L. Quo, Scientific Reports (2012).
[4] Y. Hou, J.-S. Kim, S.-W. Huang, S. Ashkenazi, L. J. Quo, and M. O'Donnell, IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control **55**, 1867 (2008).
[5] E. Biagi, F. Margheri, and D. Meichelli, IEEE transactions on ultrasonics, ferroelectrics, and frequency control **48**, 1669 (2001).
[6] T. Buma, M. Spisar, and M. O'Donnell, Applied Physics Letters **79**, 548 (2001).
[7] Y. Hou, J.-S. Kim, S. Ashkenazi, M. O'Donnell, and L. J. Guo, Applied Physics Letters **89**, 093901 (2006).
[8] A. H. Pfund, J. Opt. Soc. Am. **23**, 375 (1933).
[9] D. M. Mann and H. P. Broida, Journal of Applied Physics **44**, 4950 (1973).
[10] L. Harris, R. T. McGinnies, and B. M. Siegel, J. Opt. Soc. Am. **38**, 582 (1948).
[11] D. J. Advena, V. T. Bly, and J. T. Cox, Appl. Opt. **32**, 1136 (1993).
[12] J. Lehman, E. Theocharous, G. Eppeldauer, and C. Pannell, Meas. Sci. Tech. **14**, 916 (2003).
[13] T. A. Witten and L. M. Sander, Phys. Rev. Lett. **47**, 1400 (1981).
[14] L. Harris, *The Optical Properties of Metal Blacks and Carbon Blacks*, Massachusets Institute of Technology, 1967.
[15] F. Gao, X. Feng, L. Bai, R. Zhang, S. Liu, R. Ding, R. Kishor, Y. Zhao, and Y. Zheng, arXiv:1602.07894 (2016).
[16] F. Gao, X. Feng, Y. Zheng, and C.-D. Ohl, Journal of Biomedical Optics **19**, 067006 (2014).
[17] R. G. Stearns and G. S. Kino, IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control **34**, 179 (1987).
[18] Y. Li, X. Jiang, B. Liang, J.-c. Cheng, and L. Zhang, Phys. Rev. Applied **4**, 024003 (2015).

Bibliography

[1] T. A. Witten and L. M. Sander. "Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon". In: *Physical Review Letters* 47 (1981), p. 1400. DOI: https://doi.org/10.1103/PhysRevLett.47.1400.

[2] Charles Kittel and Herbert Kroemer. *Thermal Physics (Second Edition)*. New York: Freeman, 1980.

[3] Eli Wolf. "Dynamic Thermal Properties of Nanostructured Films". PhD thesis. University of California, Santa Barbara, 2015.

[4] Krutik Patel. "Broadband Optical Ultrasound Generation and Beamforming with Diffusion Limited Aggregates". PhD thesis. University of California, Santa Barbara, 2015.

[5] Mathias Fink. "Time reversal of ultrasonic fields. I. Basic principles". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 39 (1992), pp. 555–566. DOI: 10.1109/58.156174.

[6] LunarAccents Design Corporation. *Electrical Efficiency LED Lighting*. web. http://www.lunaraccents.c electrical-efficiency-LED-lighting.html.

[7] *Cree XLamp XP-E LEDs Product Family Data Sheet*. The relevant part number is XPEWHT-L1-0000-00AE7. Cree.

[8] S.J. Frank L. Pedrotti, Leno S. Pedrotti, and Leno M. Pedrotti. *Introduction to Optics (Third Edition)*. Upper Saddle River, NJ 07458: Pearson Prentice Hall, 2007.

[9] R. G. Stearns and G. S. Kino. In: *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control* 34 (1987), p. 179.

[10] E. Biagi, F. Margheri, and D. Menichelli. "Efficient laser-ultrasound generation by using heavily absorbing films as targets". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 48 (6 2001), pp. 1669–1680. DOI: 10.1109/58.971720.

[11] H. W. Baac et al. In: *Scientific Reports* (2012).

[12] Newport Corporation. *Technical Note: Gaussian Beam Optics*.

[13] F. Joseph Pompei. "Sound From Ultrasound: The Parametric Array as an Audible Sound Source". PhD thesis. Massachutsetts Institute of Technology, 2002.

[14] Athanasios Karamalis, Wolfgang Wein, and Nassir Navab. "Fast Ultrasound Image Simulation using the Westervelt Equation". In: *Medical Image Computing and Computer Assisted Intervention* (2010).

[15] T. Buma, M. Spisar, and M. O'Donnell. "High-frequency ultrasound array element using thermoelastic expansion in an elastomeric film". In: *Applied Physics Letters'* 79 (2001), p. 548. DOI: 10.1063/1.1388027.

[16] Yang Hou et al. "Characterization of a broadband all-optical ultrasound transducer-from optical and acoustical properties to imaging". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 55 (8 2008). DOI: 10.1109/TUFFC.2008.870.

[17] H. Usui, I. Yamada, and T. Takagi. "Anthracene and polyethylene thin film depositions by ionized cluster beam". In: *Journal of Vacuum Science & Technology A* 4 (1986), p. 52.

[18] Hiroaki Usui et al. “Characteristics of Polyethylene Thin Films Deposited By Ionized Cluster Beam”. In: *Mat. Res. Soc. Symp. Proc.* 10 (1988).

[19] Peter G. Bruce, Bruno Scrosati, and Jean-Marie Tarascon. “Nanomaterials for Rechargeable Lithium Batteries”. In: *Angewandte Chemie* (2008). DOI: 10.1002/anie.200702505.

[20] Sofia Magkiriadou. “Structural Color from Colloidal Glasses”. PhD thesis. Harvard University, 2014.

[21] Robert J. Westervelt. “Scattering of Sound by Sound”. In: *The Journal of the Acoustical Society of America* (1957).