DESIGN AND FABRICATION OF A HOLOGRAPHIC DIFFRACTIVE OPTICAL ELEMENT

by

YIYUN WU

A Thesis Submitted to the Faculty of the

WYANT COLLEGE OF OPTICAL SCIENCES

In Partial Fulfillment of the Requirements for the

Degree of MASTER OF SCIENCE

in the Graduate College

The University of Arizona

Tucson, Arizona

2024

Copyright © YiYun Wu, 2024

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Master's Committee, we certify that we have read the thesis prepared by **YiYun Wu,** titled ***Design and Fabrication of a Holographic Diffractive Optical Element*** and recommend that it be accepted as fulfilling the thesis requirement for the Master's Degree.

Thomas D Milster
Professor Thomas D. Milster Date: 4/25/2024

Yuzuru Takashima
Professor Yuzuru Takashima Date: 4/25/2024

Professor Young Sik Kim Date: 4/25/2024

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Graduate College.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the Master's requirement.

Professor Thomas D. Milster Date: 4/25/2024
Master's Thesis Committee Chair
Wyant College of Optical Sciences

Acknowledgements

Thank you to Tom Milster for your invaluable guidance throughout this research project.

Nick Brar, for showing me the ropes and allowing me to master lab equipment needed to complete this project on such short notice.

Youngsik, Edgar, and Zichan, for being such welcoming and supportive lab mates.

Eric, for asking insightful questions and motivating me to deliver at a higher quality.

Chris, Takuto, Brian, Donnis, and Goldie at Nikon Research Corporation of America.

Kira, for doing the process development and characterization work that made my work in this project possible.

Jeremy and Calvin for fixing the Thorlabs camera I broke 2 months into the project.

Zhihan, for always being available on short notice to help me measure the index of samples.

Joe and Roland, for training me on the Versaline and showing me endless patience when I made mistakes handling complicated machinery in the cleanroom. Thank you for always being on hand to help me when I ran into issues.

Shelby, for being such a brilliant mentor and for getting me interested in holography as an undergraduate student.

Prof. Takashima, for teaching a wonderful course on diffractive optics, which served as excellent preparation for this project.

Jon, for going out of your way to help me find MATLAB optimization methods that worked for this project.

I'd like to acknowledge that the Plasma-Therm Versaline reactive ion etcher used for this project is obtained through an NSF grant with award number ECCS-1725571. This project could not have been possible without it.

Dedication

This thesis is dedicated to my family: mom, dad, and my brother Jhih-Yang. I carry you all in my heart everywhere I go.

Contents

List of Tables

List of Figures

Acronyms

AU – Address Unit

CAD – Computer-Aided Design

CGH – Computer-Generated Hologram

DE – Diffraction Efficiency

DI – De-ionized

DMD – Digital Multi-mirror Device

DOE – Diffractive Optical Element

DXF – Design Exchange Format

FS – Fused Silica

G-S – Gerchberg-Saxton

HF – High Frequency

HPC – High-Performance Computer

ICP – Inductively Coupled Plasma

IMTEK - Institut für Mikrosystemtechnik

MLA – Maskless Aligner

NRCA – Nikon Research Corporation of America

OPD – Optical Path Difference

PFPE – Perfluoropolyether

PR - Photoresist

PTFE – Polytetrafluoroethylene

PV – Peak-to-valley

RGB – Red, Green, Blue

RIE – Reactive Ion Etcher

ROI – Region of Interest

RONI – Region of No Interest

RPM – Rotations Per Minute

SCCM – Standard Cubic Centimeter per Minute

SNR – Signal-to-Noise Ratio

Abstract

This thesis presents the design and fabrication of a holographic diffractive optical element (DOE) similar in concept to a Dammann grating that produces an array of uniformly spaced laser beams of equal irradiance. To produce an angular offset between the 0th order and the array of higher order beams, two-dimensional computer-generated hologram (CGH) design techniques are employed. Starting with a base Gerchberg-Saxton algorithm, modifications are introduced to factor in the source distribution and to make the array of beams more uniform in irradiance. An initial design, fabrication, and test phase is carried out using a 650 nm source to allow for rapid on-site design iteration cycles and the timely correction of unforeseen issues before samples are fabricated for a 1064 nm source. Fused silica etch process development is done in parallel with the design iterations. Preliminary samples are fabricated using pre-established grayscale photolithography procedures, and the final sample is etched into fused silica to be used in an undisclosed high-power application.

Chapter 1

Introduction

1.1 Dammann Gratings

This thesis project involves the design and fabrication of a Dammann-style grating, or a transmission phase modulation grating that produces a uniform grid of laser beams [2]. In a seminal paper by Dammann and Görtler published in 1971 [4], binary phase grating structures are shown to be able to generate an array of laser beams that are uniform in irradiance. As shown in the figure below, the duty length of the binary structures can be tuned to produce a specified number of diffraction orders.

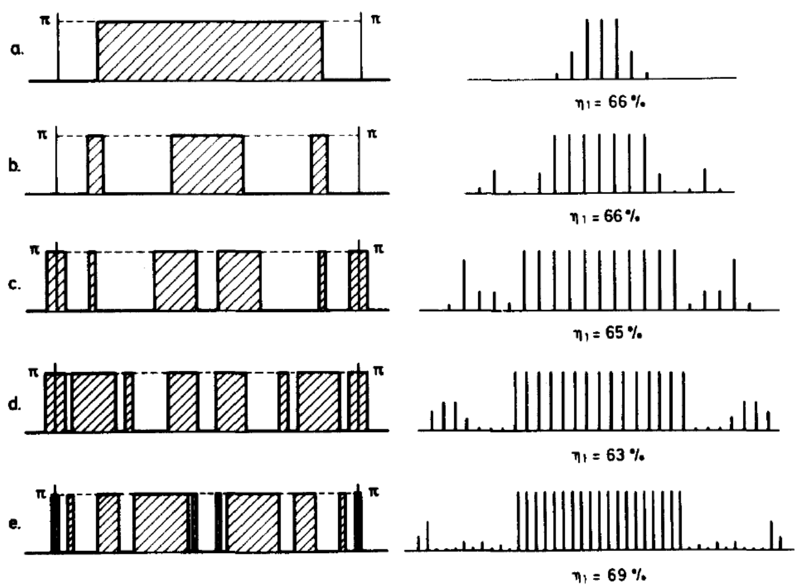


Figure 1. Dammann and Görtler 1971 [4]. Binary phase structures that generate 3, 7, 11, 15, and 19 diffraction orders of uniform irradiance

While binary phase gratings are simpler to fabricate, note that the binary gratings have diffraction efficiencies of less than 70%. Dammann and Görtler show that when multi-level structures that approach continuous groove-shapes are fabricated, diffraction efficiencies of up to

96% can be achieved. Diffraction efficiency in this case is the total percentage of light diffracted into the uniform orders. The maximum amount of phase modulation introduced is 2π or one wavelength. For this project, previous characterization work on a maskless photolithographic tool allowed for the fabrication of multi-level or grayscale phase structures that approach the continuous phase structures described by Dammann and Görtler [4].

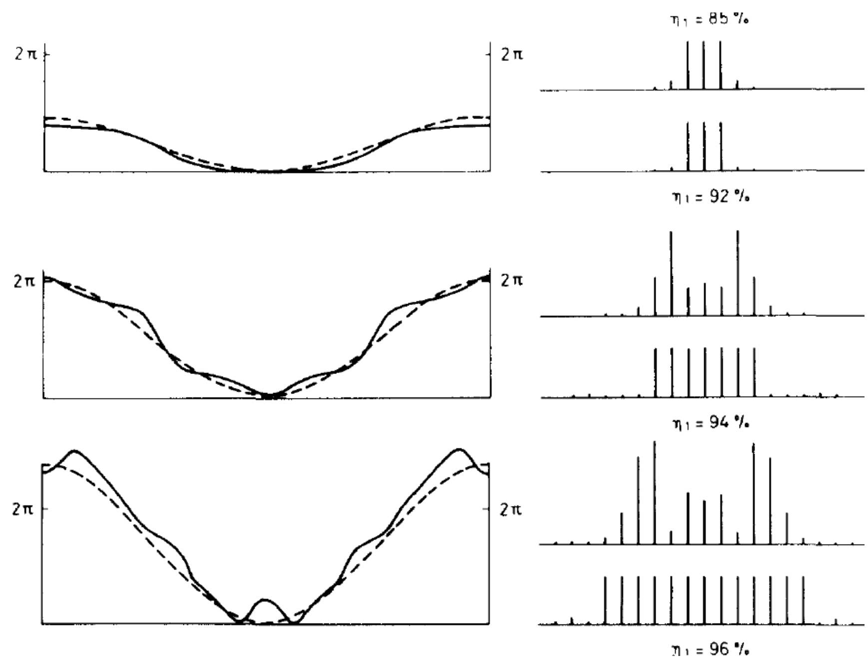


Figure 2. Dammann and Görtler 1971 [4]. Continuous phase modulation structures are used to increase diffraction efficiency. The dotted lines represent sinusoidal groove shapes.

1.2 Project Scope

This project involved the design of a DOE that produced a single array of diffracted orders, or signal orders, with an angular offset between the 0^{th} order and higher order diffracted beams, as shown in Figure 3. This angular offset, combined with the optomechanical requirement for it to be achieved using a single flat 1-4 mm thick element, called for the use of a two-dimensional holographic pattern.

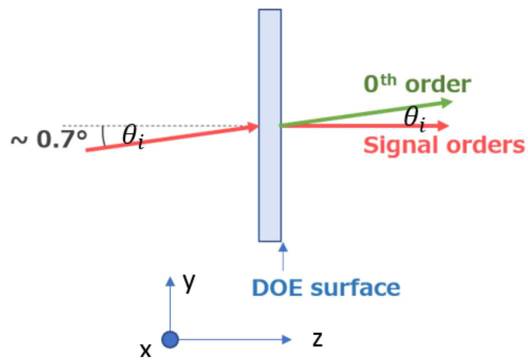


Figure 3. Schematic showing intended implementation of DOE

Note that the groove patterns described by Dammann and Görtler in 1971 are one-dimensional patterns because the diffractive grooves run vertically across the active area of the fabricated DOE. A two-dimensional phase grating can also be referred to as a computer-generated hologram (CGH) because it physically stores the phase information necessary to recreate a specified image. The term "computer-generated" derives from the fact that these hologram designs are produced digitally, without the use of interference-based encoding techniques used in traditional holography. The advent of CGH design as well as lithography-based microfabrication has opened the door for increasingly precise wavefront modulation useful for a host of applications.

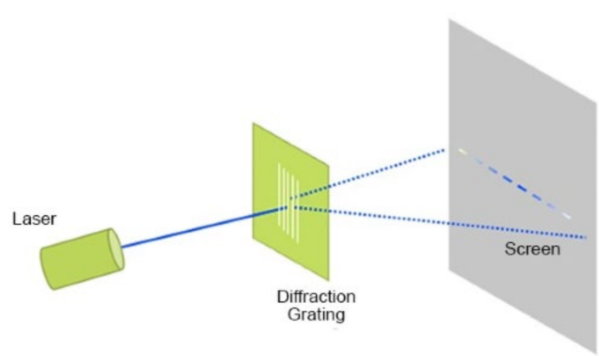


Figure 4. Example of a one-dimensional grating, which is not in the scope of this project. Figure courtesy of GoPhotonics.com [2]

1.3 Project Overview

For the purposes of design and simulation, it is assumed that the laser beam is normally incident on the DOE surface and the image plane is designed to include this angular offset in signal orders. Figures 3 and 5 show that the 0^{th} order is simply undeviated light that goes through the DOE and does not get diffracted in the direction of the signal orders. If the CGH is fabricated to have the correct depth, the amount of light into the 0^{th} order will approach the ideal of zero percent.

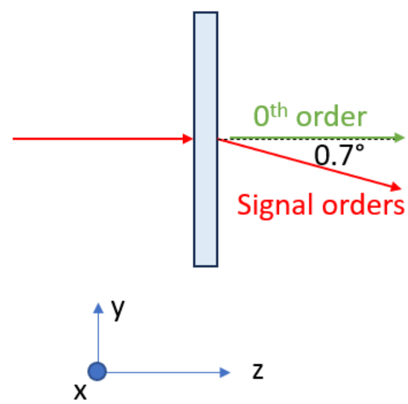


Figure 5. Schematic showing configuration assumed for the purposes of DOE design

To design the DOE, a bitmap file is created to represent the desired image plane, as shown in Figure 6. This bitmap file is then fed through a Gerchberg-Saxton algorithm, which utilizes a series of Fourier transforms to translate back and forth between the desired image plane and the phase map plane or CGH plane. Constraints on irradiance are placed on each subsequent Fourier transform, allowing the algorithm to find a phase map that corresponds to the desired image plane. This phase map is then translated into etch depths on the order of tens of nanometers when fabricated. Details of the design process is presented in Chapter 2.

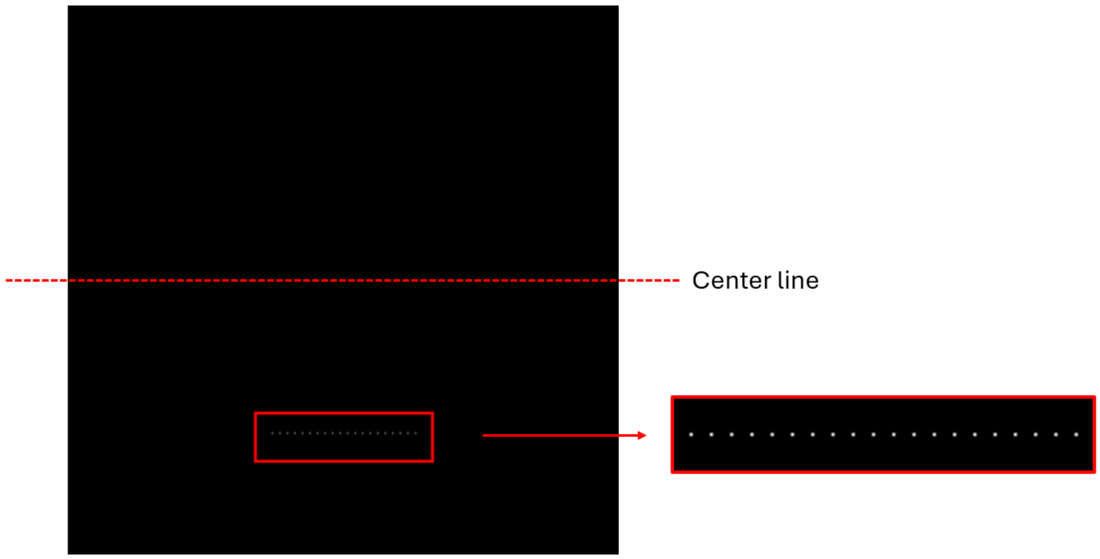


Figure 6. Specially designed image plane with 20 signal orders offset from center line

The phase map consists of matrix of grayscale values ranging from 0 to 255, with higher values representing deeper etch depth and therefore less phase delay introduced. When a collimated laser beam is incident on the phase modulation surface, or the DOE surface, each point on the phase map introduces a different amount of phase delay to the incident wavefront, thus precisely reshaping the wavefront so that the desired array of laser beams is produced in the image plane. For the purposes of fabrication, grayscale values are translated to exposure dosages in a maskless photolithography machine [13]. Since the CGH is fabricated with a positive photoresist, areas that are exposed to a high dosage of light correspond to areas of greater etch depth after development. Details of the fabrication process is presented in Chapter 3.

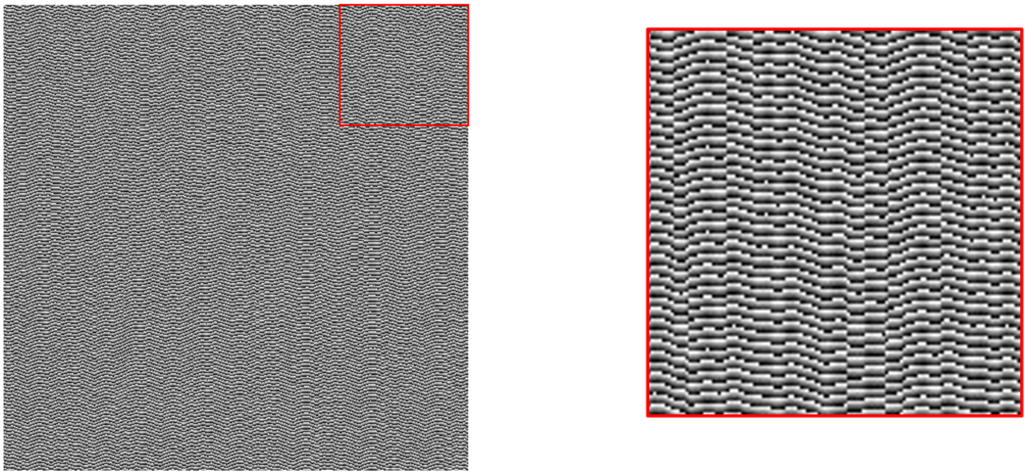


Figure 7. (Right) A 512x512 phase map. (Left) Closeup of phase map.

1.4 Design Specifications

The initial specifications for the DOE are displayed in Table 1, which are for the desired angular separation of signal orders produced by the DOE.

Specification	Value/ Requirement	Tolerance	Notes
Diffraction Angles	0^{th} order remains θ_i from normal +/- 10 signal orders emerge 0° from normal; orders separated by 0.035°	Tolerance in θ_i +/- 5 % of 0.035°	Signal orders are odd orders
Angle of incidence θ_i	0.7°	Can accept 0.6° to 1°	**Design for 0° incidence, operate at 0.7°**

Table 1. Imaging Specifications

Figure 8 shows the desired 0.7-degree offset between the 0^{th} order beam and the array of signal order beams. The desired angular separation between signal order beams is 0.035 degrees. The DOE is designed to maximize light into the 20 signal orders, while minimizing light into the higher orders and the 0^{th} order. Since the 20 signal orders are in the position commonly associated with the +/- 10 odd diffraction orders, the signal orders may be referred to as the +/- 10 orders in this thesis.

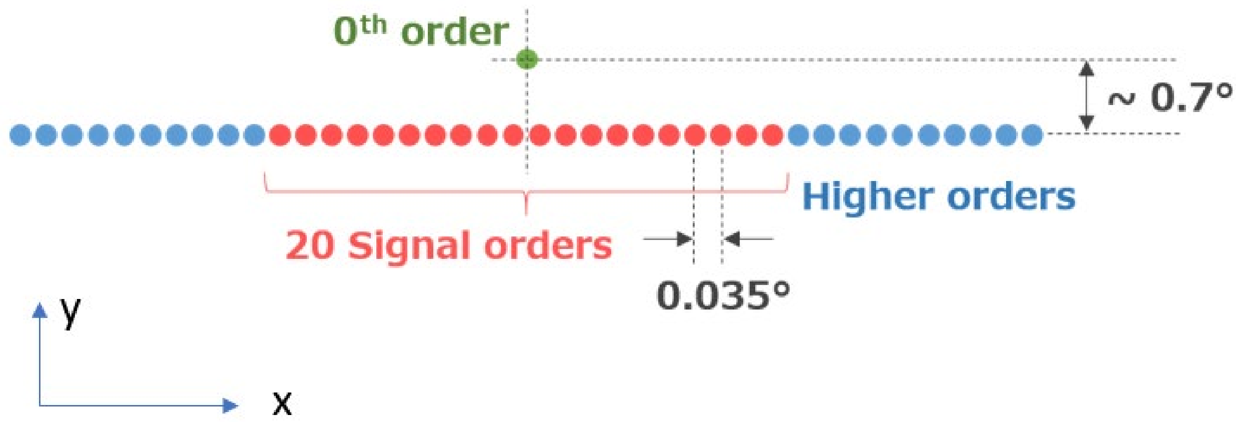


Figure 8. Desired angular separation between diffracted orders

In addition to the main specifications regarding the angular separation of generated laser beams, goals were also established regarding the *absolute* diffraction efficiency, which is defined as the percentage of light incident on the DOE that gets diffracted into the 20 signal orders. The peak-to-valley (PV) uniformity is the peak-to-minimum signal power divided by the average signal power. The ideal PV uniformity is for each of the 20 signal orders to have equal irradiance, or to have 0% PV uniformity. That is,

$$Uniformity = \frac{\max(relative\ DE) - \min\ (relative\ DE)}{mean(relative\ DE)} \tag{1}$$

Another parameter relevant to this project is the *relative* diffraction efficiency, which is defined as the percentage of light that is diffracted into one of the 20 signal orders. The ideal relative diffractive efficiency is 5%, since 100 divided by 20 is 5.

Table 2 shows other specifications that are relevant to the project, including the operation wavelength of 1064 nm and the $\frac{1}{e^2}$ beam diameter. These parameters have a major bearing on the design of the DOE. Throughout the prototyping process, the phase map designs are only etched in photoresist to allow for a rapid design and test cycle. However, the final phase map design is etched in fused silica since the desired application for this DOE involves a high-power laser that could vaporize photoresist.

Specification	Value/ Requirement	Tolerance	Notes
Grating type	Phase modulation	-	-
Wavelength	1064 nm	-	-
Substrate	Fused silica	-	From Edmund Optics (EO)
Substrate Thickness	1 - 4 mm	-	Spec from EO = 2 mm
Etch Depth	Depends on design	+/-5% from as-designed etch depth	-
Coating	AR	-	To be out-sourced.
Diffraction Efficiency	> 96% into +/- 10 odd orders	-	-
Uniformity	< 10%	-	PV/mean
Beam Diameter $[-\frac{1}{e^2}, \frac{1}{e^2}]$	13.4 mm	-	Confirmed by sponsor

Table 2. DOE Performance Specifications

1.5 Project Phases

The first phase of this project involves design and fabrication at 650 nm. Since a 650 nm laser is readily available, initial rapid design and testing cycles at this visible wavelength allow for the correction unforeseen challenges and the collection of a proof-of-concept result before preparing samples to test at 1064 nm.

Chapter 2

CGH Design

2.1 Pixel Size Calculations

The pixel size or address unit (AU) of a fabricated CGH design determines the angular width of the reconstructed image, like how the grating period of a diffraction grating determines the diffraction angle of diffraction orders. It can be observed that the form of the equation for AU is like the diffraction equation, as shown below in Eq. (2) below, where α is the diffraction angle, m is the diffraction order, λ the source wavelength, and Λ the grating period.

$$\sin(\alpha) = \frac{m\lambda}{\Lambda} \tag{2}$$

The CGH image is designed in angle space, so it is necessary to define a parameter called θ_{max} that represents the full angular subtense of the image produced by the hologram. Half of θ_{max} corresponds to the diffraction angle α. The diffraction order m is set to one, and 2 times the AU corresponds to the grating period Λ. To understand intuitively why 2AU corresponds to the grating period instead of just AU, one must remember that a single pixel by itself cannot introduce optical path difference. The width of two adjacent pixels of differing etch depths is the smallest feature on the CGH that introduces differential phase delay and contributes to the diffraction pattern. In Chapter 4, it is noted that omitting this factor of 2 in the equation caused the hologram image for the first fabricated design to be incorrectly scaled by a factor of 2.

$$\sin\left(\frac{1}{2}\theta_{max}\right) = \frac{\lambda}{2AU} \tag{3}$$

Since the CGH to be designed and fabricated in this project functions like a diffraction grating, it is necessary to perform calculations to ensure that the hologram image would occupy the correct angular extent. This calculation is done by first determining the AU and deciding the total CGH pixel count. For example, for a 13.4 mm diameter beam, an appropriate CGH size would be 14 mm by 14 mm, which can be achieved with an AU of 14 microns and 1000x1000 pixels. For an AU of 14 microns, θ_{max} is 4.36 degrees from Eq. (3). The desired image plane is decomposed into 1000x1000 pixels as well, where each pixel represents a $\Delta\theta$ of $\frac{4.36}{1000} = 0.00436$ degrees. A 1000x1000 pixel bitmap representing the target image is then created, where the locations of diffraction orders are set to the maximum 8-bit integer grayscale value 255, and the rest of the canvas is populated with zeros. Subsection 2.2 describes how this bitmap file is used as a constraint in design algorithm to produce the CGH design.

Table 1 specifies that the separation between the array of signal orders and the 0^{th} order be 0.7 degrees, which is 141.08 times $\Delta\theta$. In creating the target image file, this value is necessarily rounded to 141, which yields a 0.06% error. Additionally, Table 1 specifies a 0.035 degree separation between signal orders, which is 7.054 times $\Delta\theta$. Rounding to 7 yields a 0.77% error. Note that these errors are due to $\Delta\theta$ not being a factor of the intended diffraction angle specifications. Diffraction angle errors of less than 5% are considered acceptable. Since $\Delta\theta$ is determined by both the AU and the CGH size, these two design variables need to be specially chosen to minimize diffraction angle error. Throughout the project, the AU and CGH size are changed with each design iteration, but the above calculations provide an example of the calculations that are necessary to design a CGH that is large enough for the beam size while minimizing diffraction angle error.

VARIABLES				
Wavelength	1.06E-06	[m]		
AU	14	[um]		
CGH size	1000	[px]		
CALCULATIONS				
CGH size	14	[mm]		
theta_max	4.36	[deg]		
delta_theta	0.004356	[deg]		
SIGNAL ORDER LOCATIONS		Rounded	Error (deg)	% Error
0 to signal	160.72	161	0.0012	0.18%
Order separation	8.04	8	0.0002	0.45%
-10 to 10	152.68	153	0.0014	0.21%

Table 3. Screenshot of design spreadsheet showing predicted diffraction angle errors in red for a CGH with AU = 14 microns, 1000x1000 pixels, operating at 1064 nm.

2.2 Gerchberg-Saxton Algorithm

The Gerchberg-Saxton phase retrieval algorithm is used to find the hologram design using a specially designed image plane as the constraint. Figure 9 shows the flow of the algorithm, starting with a random phase initialization and a constraint on irradiance based on the desired image plane, or I_0 in the figure. Taking the Fourier transform allows for translation to the pupil plane, which in our case is the CGH plane. In the CGH plane, the amplitude information is discarded and replaced with irradiance constraints based on the source distribution, which is I_s in the flowchart below. Another Fourier transform is taken to return to the image plane, where the amplitude is once again discarded and constraints on the irradiance of the image plane are once again applied [17]. This iterative process allows the algorithm to find the phase map corresponding to a given image.

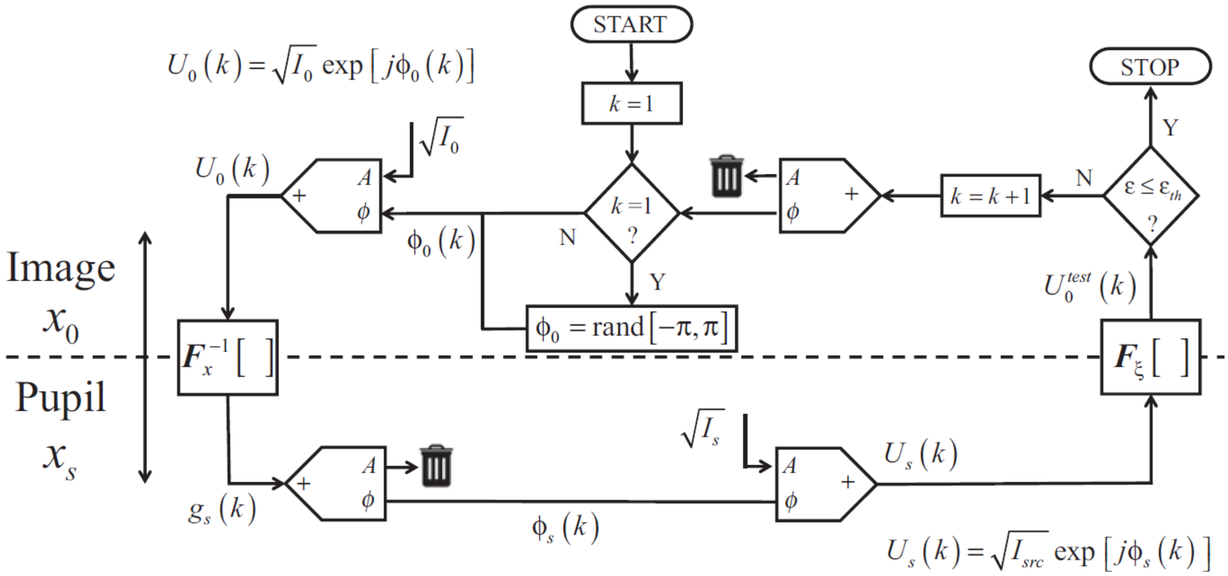


Figure 9. Flowchart of Gerchberg-Saxton algorithm [17]

One can determine when to end the algorithm by setting a threshold ϵ_{th} for the error between the CGH-generated image and the ideal image. For this project, however, the algorithm is stopped at a specified maximum number of iterations. In the figure below, one can observe that the signal-to-noise ratio, which is a measure of how closely the image produced by the CGH matches the ideal image, increases non-linearly and converges to about 10.75 dB after 100 iterations of the Gerchberg-Saxton algorithm.

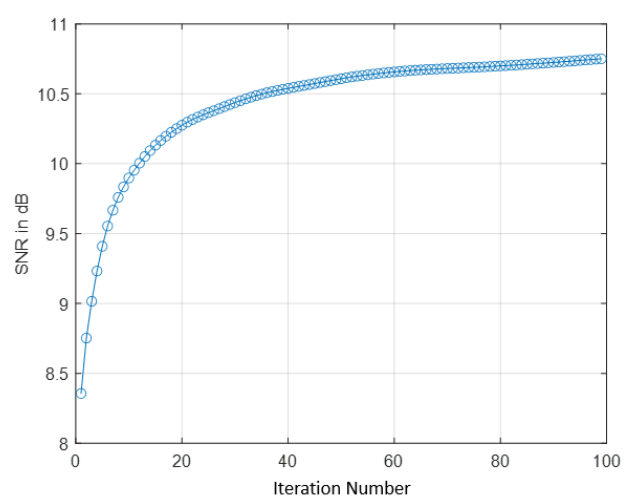


Figure 10. Convergence of Gerchberg-Saxton algorithm after 100 iterations

2.3 Optimization for Beam Uniformity

Like most design algorithms, the starting point of the Gerchberg-Saxton algorithm has a major bearing on the output of the algorithm. Since the Gerchberg-Saxton algorithm uses random phase initializations as its starting point, it is necessary to run the algorithm a thousand or more times with different random starting points to find the design that meets the design specifications. The main specification being optimized for is beam uniformity, which is a measure of how even the light is diffracted into each of the +/- 10 orders is. Without undergoing fabrication, an estimate of the uniformity of a given CGH design is calculated by converting the quantization values in the range of 0 to 255 to phase values, and taking the Fourier Transform to find the field at the image plane. Then one gaussian mask is placed on each of the +/- 10 signal orders, and the irradiance values under each mask are summed and divided by the sum of all values across the image plane. This procedure results in the relative diffraction efficiency of each diffraction order, where the ideal relative diffraction efficiency is 5% (or 0.05 in the bar graphs). In the figure below, the left bar graph shows a close-to-optimum result, which is the generation of a uniform array of diffracted orders. In the right bar graph, one observes that while high diffraction efficiency is desirable, the unevenness in irradiance across the array disqualifies the design from consideration for fabrication.

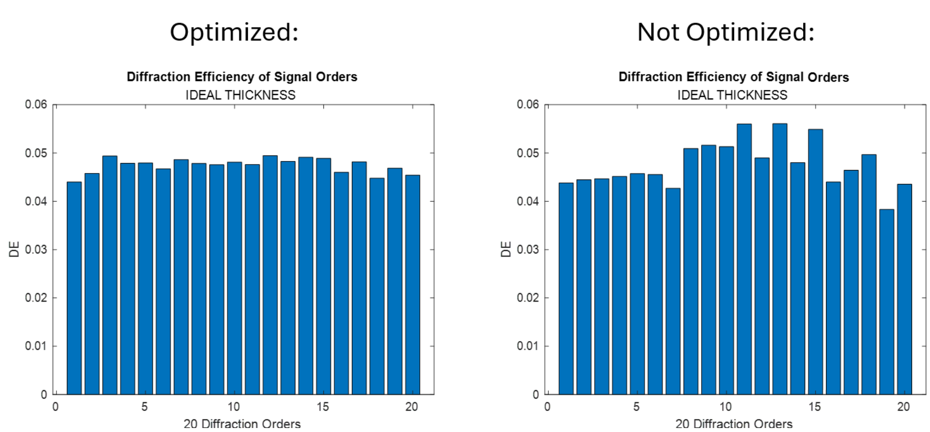


Figure 11. An "optimized" design (left) would produce a uniform array of laser beams

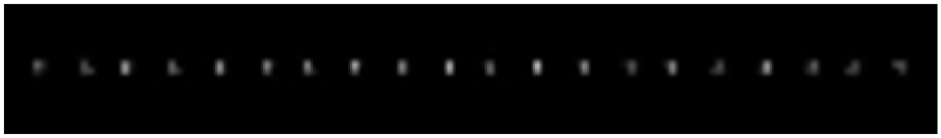


Figure 12. Simulated example of a non-uniform array of beams

2.4 Merit Function

In initial rounds of design and fabrication, the MATLAB software (see Appendix C) simply chose the "best" design by finding the design with the lowest uniformity error. Recall that uniformity is calculated as the peak-to-valley difference in diffraction efficiency divided by the mean relative diffraction efficiency. Later, it was determined that the introduction of a merit function to assist with the selection of CGH designs would be useful. For each CGH design produced by the Gerchberg-Saxton algorithm, values of the mean relative diffraction efficiency, normalized standard deviation, uniformity, and absolute diffraction efficiency are calculated and stored in a vector. A weighted root sum squared merit function is calculated for each design, with higher weights for absolute diffraction efficiency and uniformity. The normalized standard deviation is the standard deviation of relative diffraction efficiencies divided by the mean relative diffraction efficiency. A design with a low normalized standard deviation corresponds to a design with desirable uniformity.

In the figure below, uniformity and diffraction efficiency data from 470 CGH designs have been plotted, and the design chosen by the merit function is shown with a data label in the bottom right corner. The chosen design has a diffraction efficiency of 82% and a uniformity of 12%. The scatterplot shows that the merit function has correctly selected the most optimal design based on highest diffraction efficiency and lowest uniformity.

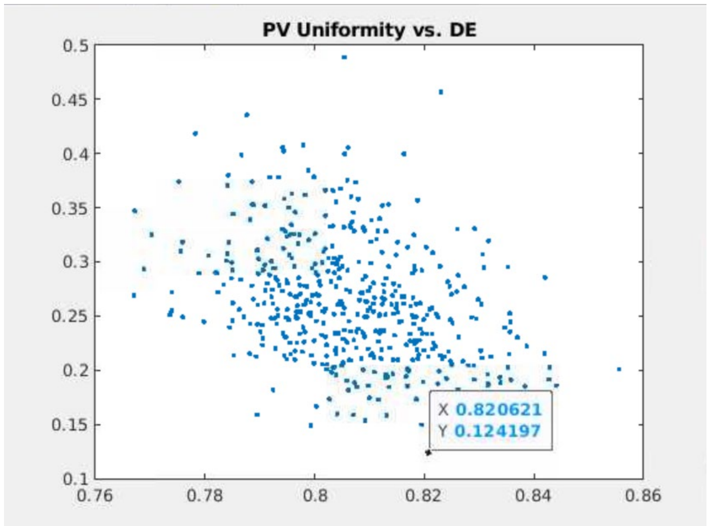


Figure 13. The merit function has selected the design with the data label

2.5 Simulation of Non-Ideal Etch Depth

In section 1.4, a +/-5% etch depth tolerance is specified, which implies that the CGH can be fabricated to within 5% of its ideal etch depth using pre-established grayscale photolithography processes [13]. The maximum-OPD etch depth is defined as the difference between the unetched surface of the CGH and the deepest feature of the CGH. Ramp designs are included on the edge of a CGH pattern area to make this parameter easy to access and measure after fabrication. The equation for calculating the max-OPD etch depth is displayed in Eq. (4), where n_2 is the index of the CGH and n_1 is the index of the object space, which is nominally 1.

$$h = \frac{\lambda}{n_2 - n_1} \tag{4}$$

To find the ideal max-OPD etch depth in photoresist at 650 nm, n_2 is 1.62 since that value is the index of the photoresist of at the specified source wavelength, and the max-OPD etch depth is found to be 1.0484 μm. To simulate a +/-5% deviation from the ideal etch depth, one simply needs to convert the change in depth to a change in phase using Eq. (5). In this formula, both $\Delta\phi$ and Δh are in fractions, where $\Delta\phi$ is a fraction of 2π and Δh is a fraction of the ideal etch depth.

$$\Delta\phi = \frac{\Delta h\ (n_2 - n_1)}{\lambda} \tag{5}$$

To derive this formula, one first notes that the amount of optical path difference (OPD) introduced by a point in the CGH is directly proportional to the amount of phase delay introduced. The equation for OPD given below is for a fraction of a wavelength. This is useful for CGH design because CGH's are often used to introduce OPD in a given incident wavefront in the zero to one wave range. Since the CGH will operate in air, n_1 is 1 and n_2 is the index of the DOE surface, which for much of the project was the index of photoresist.

$$OPD = \frac{(n_2 - n_1)h}{\lambda} \tag{6}$$

$$\phi = \frac{(n_2 - n_1)h}{\lambda} 2\pi\ [radians] \tag{7}$$

$$\frac{\partial OPD}{\partial h} \propto \frac{\partial \phi}{\partial h} \tag{8}$$

Taking the derivative of the OPD equation with respect to etch depth *h*, the following is found:

$$\frac{\partial OPD}{\partial h} = \frac{(n_2 - n_1)}{\lambda} \tag{9}$$

Solving for ΔOPD yields:

$$\Delta OPD = \frac{\Delta h\ (n_2 - n_1)}{\lambda} \tag{10}$$

Differential phase can be converted from units of radians to a fraction of 2π, in which case differential phase and differential OPD are equal, and Eq. (5) is found. This equation is used to find the phase difference that corresponds to the 5% tolerance in etch depth. This phase difference can be applied to a given CGH design to simulate the effects of this tolerance in etch depth. The result found is that deviating from the ideal etch depth decreased diffraction efficiency, but the ratios between the amount of light going into each order remained the same.

	-5% from Ideal	Ideal Etch Depth	+5% from Ideal
Avg. DE	4.34%	4.39%	4.34%
Std dev	0.14%	0.14%	0.14%
Std dev/mean	3.15%	3.15%	3.15%
PV uniformity	11.10%	11.14%	11.19%
Sum	86.87%	87.76%	86.88%

Table 4. Data for a 511x511 pixel CGH design with timestamp "20-Mar-2024_134226"

The above table shows that a +/-5% deviation from the ideal etch depth causes the absolute diffractive efficiency, labeled as "sum" in the table, to drop by about 1%, which is insignificant. The change in the uniformity of the beam array is also minor, which shows that a +/-5% tolerance in etch depth will not significantly degrade the performance of the CGH design.

2.6 Application of Source Distribution

In the CGH plane of the Gerchberg-Saxton algorithm, a source distribution is applied to constrain the irradiance before taking the Fourier transform to return to the image plane. For this project, the source distribution used is a Gaussian beam, and the source distribution matrix applied in the design algorithm is specially tailored to the size of the beam with respect to the physical size of the CGH. For example, for a CGH with a side length of 12.3 mm and a beam with a $\frac{1}{e^2}$ diameter of 13.4, the normalized irradiance value of the source matrix at its top edge would be 0.1854, as shown in red in Figure 14. In this way, the source distribution matrix applied in the Gerchberg-Saxton algorithm is designed to be a cutout of the full beam.

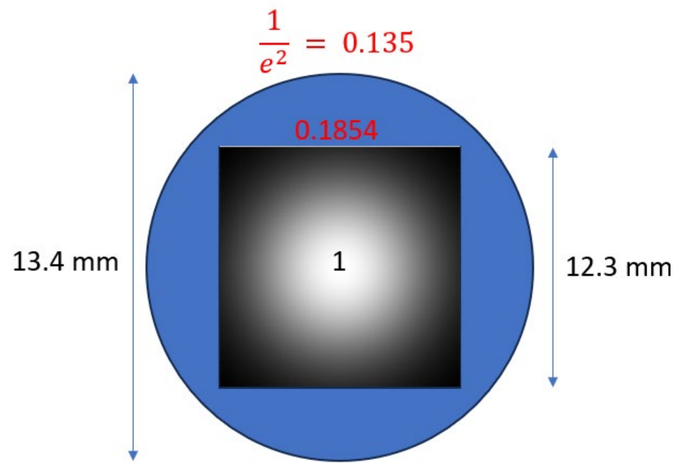


Figure 14. The blue circle represents the $\frac{1}{e^2}$ beam diameter, and an insert of the source distribution matrix is shown in grayscale. Numbers in red show the normalized irradiance at the edge of the CGH area (0.1854), and at the edge of the beam (0.135)

It is essential that the source distribution matrix be tailored to the beam to be used with the CGH since the source distribution has a major bearing on whether the actual image produced by a fabricated CGH can match the simulated image. It is found during testing cycles at 650 nm that discrepancies between source distributions used during design and the actual source distribution caused significant degradations in PV uniformity.

2.7 Region of Interest/ Region of No Interest

Besides the application of a source distribution, another modification is made to the Gerchberg-Saxton algorithm that defines a region of interest (ROI) in the target image, which is processed with the standard Gerchberg algorithm, and a region of no interest (RONI), where irradiance is not constrained. Recall that the Gerchberg-Saxton algorithm starts in the image plane with a random phase initialization. Taking the Fourier transform is equivalent to traversing to the CGH plane, where a source distribution constraint is applied before returning to the image plane. In the image plane, where before the algorithm would discard the amplitude information and reapply constraints based on the target image, with the ROI/RONI modification the amplitude information in the RONI is kept while only constraining the amplitude in the ROI. Modifying the base Gerchberg-Saxton algorithm in this way allows the final CGH design generated to reproduce the ROI of the target image more faithfully, while allowing most of the noise to be pushed into the RONI. In this project, the ROI in the target image includes the array of laser beams, while the RONI can be set to be any region that does not contain the laser beams.

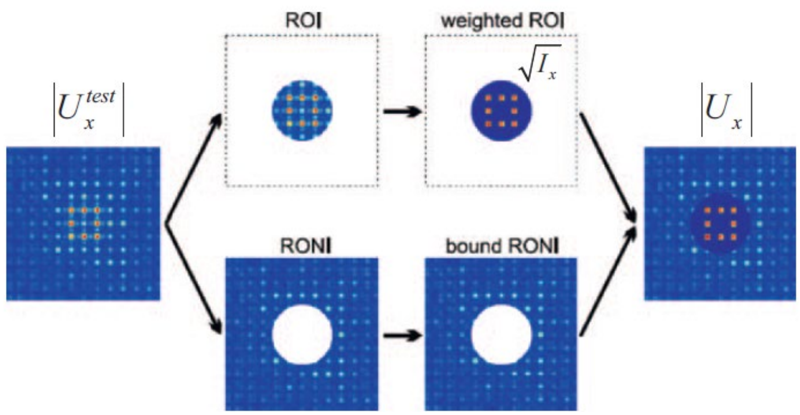


Figure 15. ROI/RONI constraining results in less noise in ROI. [1]

In Figure 15, the ROI/RONI technique used by Engström et al. [1] for a similar application is shown, where amplitude constraints are applied only in the ROI to decrease the amount of noise and increase beam uniformity within the ROI. For this project, uniformity is the main concern, so this ROI/RONI modification is used to meet the 10% uniformity specification. Since this modification allows the irradiance in the RONI to be unconstrained and noisy, a tradeoff in diffraction efficiency is found.

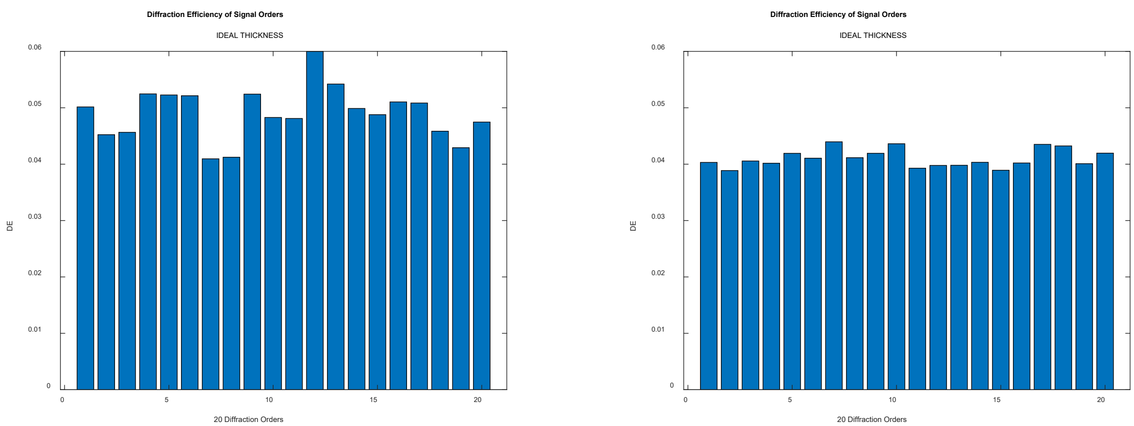


Figure 16. (left) Relative diffraction efficiency for a design found with no RONI applied. (right) Design found with RONI applied with 10% uniformity.

If a saturated detector is simulated by setting all irradiance values above a threshold to the same value, the noise in the reconstructed image of a CGH design found with ROI/RONI becomes visible. As expected, the noise is particularly high in the region of no interest, which could explain why diffraction efficiency into the +/- 10 signal orders decreased with this modification to the base Gerchberg-Saxton algorithm. Light that would otherwise contribute to non-uniformity if directed to the positions of the +/- 10 signal orders are instead scattered as noise in the image.

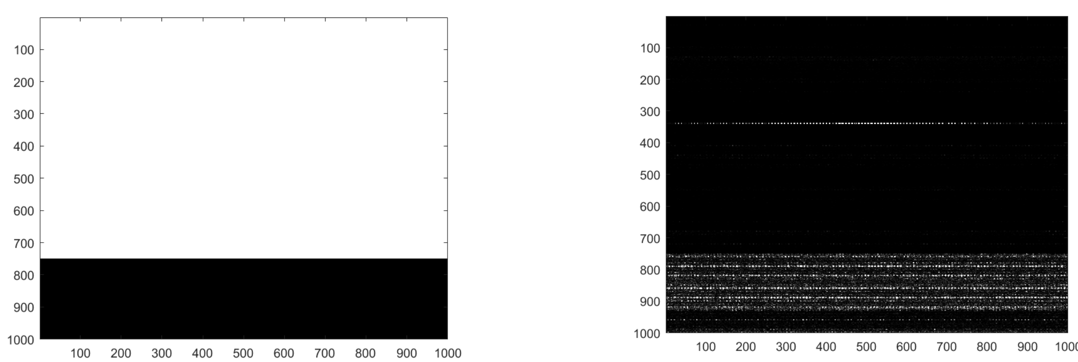


Figure 17. (left) ROI/RONI mask applied, where black represents RONI. (right) Reconstructed image from a CGH design with ROI/RONI. Noise level visible due to simulated detector saturation.

2.8 Simulated Annealing

After a CGH design is generated by the Gerchberg-Saxton algorithm, second-level optimizers can be applied to further improve the performance of the CGH design. After consulting with representatives from MathWorks, it is determined that simulated annealing optimization [16] is most suited for this project, given that the goal is to optimize two-dimensional CGH designs of high pixel counts, from 500x500 pixels for initial designs to 1000x1000 pixels for the final design. A simulated annealing optimizer simulates the physical process of heating a material and cooling the material down to reduce defects in the material. When applied to CGH optimization, the "heating" of the unoptimized CGH causes some values in the matrix to be perturbed or deviate randomly from the original value. A condition is set in the optimization algorithm to decide whether to keep the perturbed value or to restore the original value before smaller perturbations are applied to simulate "cooling." A temperature schedule can be set to control how quickly the optimizer "cools" the CGH. An objective function is called repeatedly throughout the cooling process to track changes to a variable of interest, such as uniformity. Generally, it is desirable to apply a slower cooling schedule so that the optimizer is more likely to slowly guide the original design to a local minimum.

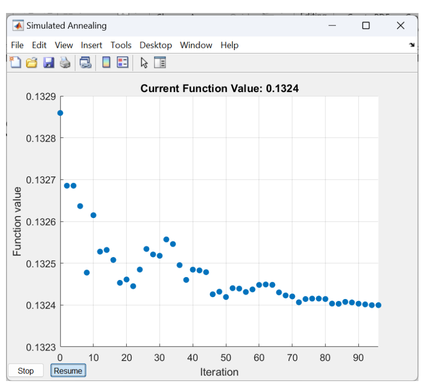


Figure 18. Uniformity plotted as a function of simulated annealing algorithm iterations. An initial value of 0.1328 is optimized to approach 0.1324.

2.9 High-Performance Computing

This chapter describes multiple design procedures that require extensive computing power, from running the Gerchberg-Saxton algorithm more than a thousand times to find a design with optimal performance specifications, to applying the simulated annealing as a second-level optimizer. These processes required running parallel computing processes on the university's high-performance computer (HPC) systems. MATLAB's built-in "parfor" command functions like the regular "for" command, but it allows each loop to be executed in parallel. Since each loop is run by an independent worker, there can be no dependencies in the information used by each worker. A typical laptop may have 4 to 6 cores, but the HPC allows for the use of a maximum of 94 cores. Running 94 loops in parallel significantly reduces the runtime of large design searches. For example, if it took the Gerchberg-Saxton algorithm 8 minutes to produce 1 CGH design, producing 1000 designs would take 133 hours. With the 94-core HPC however, it would take 8 minutes to produce 94 designs, so that time can be reduced to less than 2 hours.

$$1000\ designs * \frac{8\ min}{94\ designs} * \frac{hr}{60\ min} = 1.42\ hr \tag{11}$$

2.10 Effects of Phase Quantization

In 1970, Dallas experimentally verified that phase quantization, or the practice of decomposing continuous values within a phase map into discrete bins, causes false images to appear [19]. Figure 19 shows that the increase of quantization levels reduces the irradiance of false images. For this project, the Gerchberg-Saxton design algorithm works with continuous phase values before performing phase quantization by utilizing integer values ranging from 0 to 255. In Chapter 4, it is shown that the effects of phase quantization and non-linearity compound to produce multiple false images, thus causing a significant departure of the predicted diffraction efficiency from the actual diffraction efficiency.

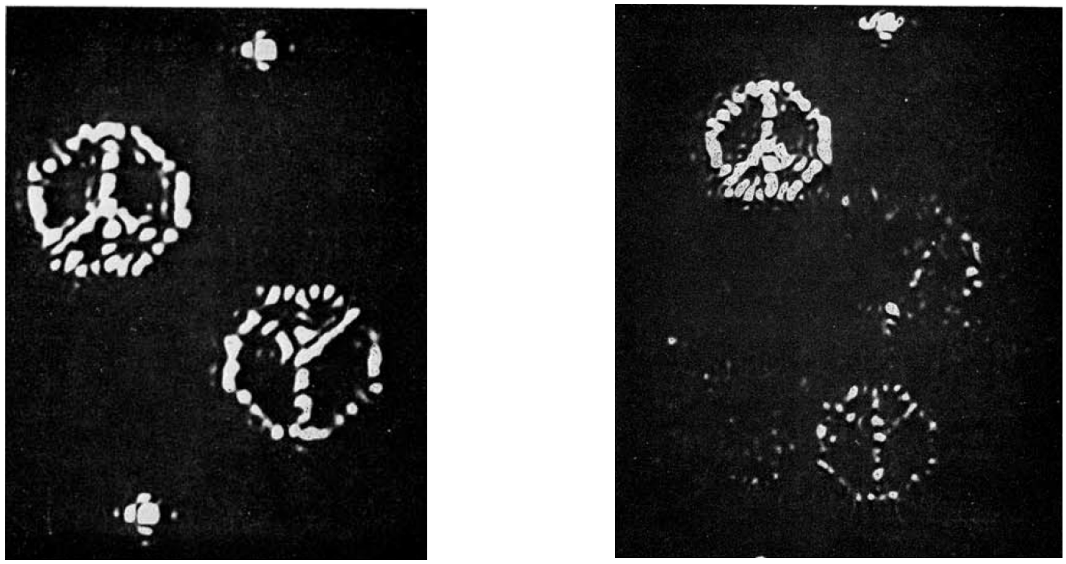


Figure 19. (left) 2-level quantization. (right) 3-level quantization [19]

Chapter 3
Fabrication

3.1 Overview

The CGH designs for this project are etched into photoresist and transferred into fused silica using well-established microfabrication techniques, thanks to process development done by past researchers. The basic steps of the fabrication process are shown in Figure 20. Photoresist is first exposed to varying dosages of light in the maskless photolithography machine. This reduces the inhibitor concentration in the photoresist by different amounts, so that when the photoresist is placed in a developer solution, the photoresist is differentially wet etched away. In this way, one can easily fabricate a given two-dimensional CGH design, which consists of a grid of pixels with varying etch depths. The product of step 2 in Figure 20 can already be tested, as is done throughout the project to provide proof-of-concept demonstrations. To transfer a CGH pattern from photoresist into fused silica, the Versaline reactive ion etcher from Plasma-Therm is used with a silicon dioxide etch recipe. The following subsections provide details on each of the fabrication steps implemented for this project.

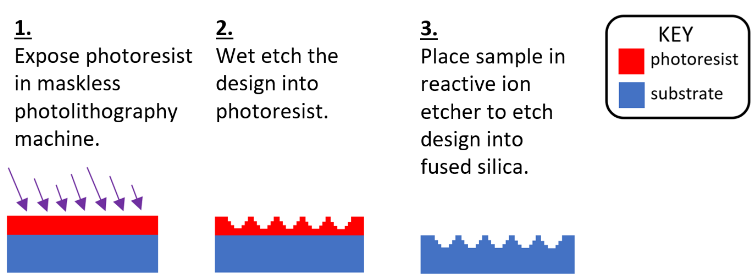


Figure 20. Main steps of CGH fabrication process

3.2 Refractive Index of Materials

Since the purpose of a CGH is to precisely reshape an incoming wavefront by introducing optical path difference differentially across its active area, the most important characteristic to consider in substrate materials is the refractive index. Throughout the course of this project, CGH designs are often first tested on photoresist spun on BK7. The reason fused silica is not used as the substrate for these initial photoresist tests is simply that BK7 is less expensive compared to fused silica. In this case, the index to be considered when calculating the designed max-OPD etch depth is in fact the index of the photoresist, and not the index of the BK7 substrate. This is because the CGH is etched in the photoresist, so an incoming wavefront gains optical path difference when traversing through the photoresist, not the BK7 substrate. As noted in Figure 21, the index of the photoresist changed depending on whether it has been bleached or not. Bleaching is accomplished by prolonged exposure to white light. Bleaching changes the absorption properties of the resist, and it is observed that after bleaching, there is less noise in the reconstructed image of a fabricated CGH. Throughout the project, the bleached index curve is used to determine the max-OPD etch depth. At 650 nm, the index of the S1827 photoresist is 1.62. Using Eq. (4) introduced in Chapter 2, it is determined that the max-OPD etch depth for CGH designs etched in photoresist operating at 650 nm is 1.048 microns.

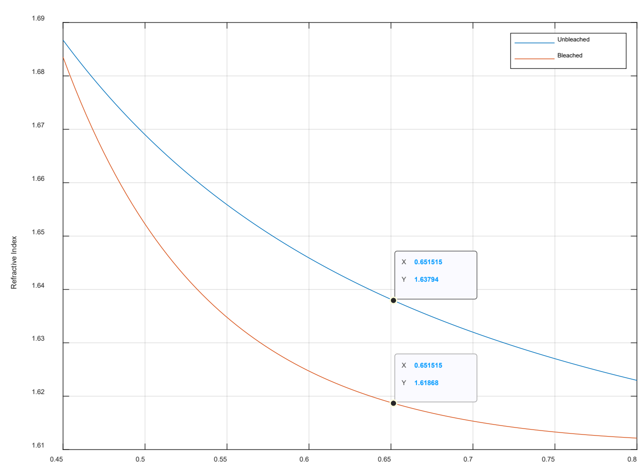


Figure 21. Index of S1827 photoresist as a function of wavelength

For the 1064 nm design and fabrication phase of the project, designs are etched into fused silica, and it is noted that the index of fused silica at the source wavelength 1064 nm is approximately 1.45, which corresponds to an ideal max-OPD etch depth of 2.364 microns.

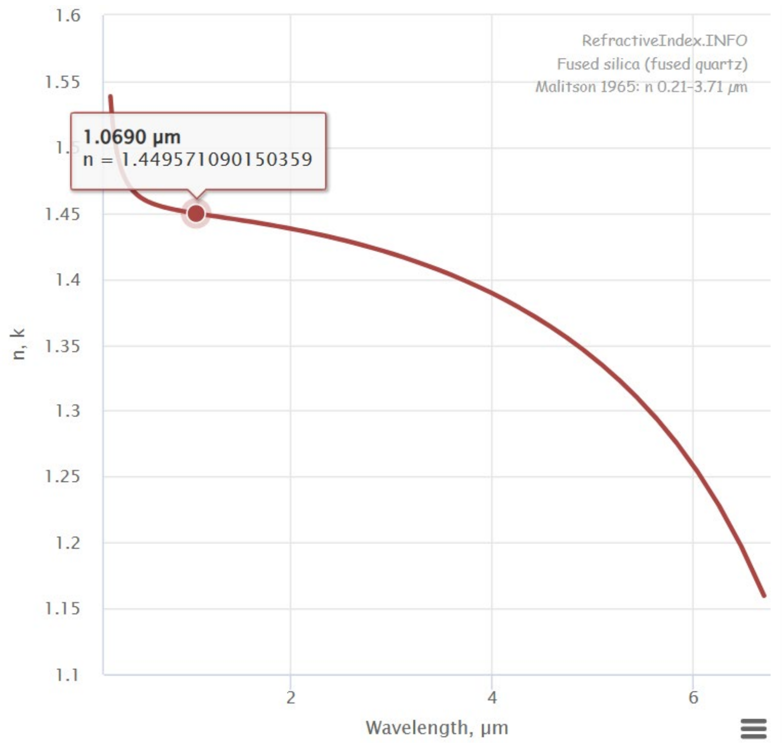


Figure 22. Index curve of fused silica [12]

3.3 Linearization

This section details how a CGH design is processed prior to fabrication. To facilitate easy measurement of the max-OPD etch depth after fabrication, test ramps are placed on the four corners.

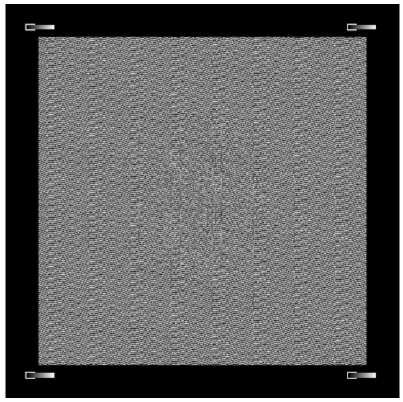


Figure 23. Bitmap of 1000x1000 pixel CGH design with ramps placed in 4 corners

The next step is to linearize the bitmap file, and this is necessary to compensate for the fact that since the photoresist responds nonlinearly to exposure dosage, there is a nonlinear relationship between exposure dosage and etch depth. To perform linearization, one first needs to obtain a calibration curve. Test ramps that cover the full range of grayscale values from 0 to 255 are fabricated. Etch profile data obtained from these test ramps are processed to produce a calibration curve that can be used to linearize the CGH. During linearization, grayscale values in the original design are mapped to new values to account for the non-linear exposure curve.

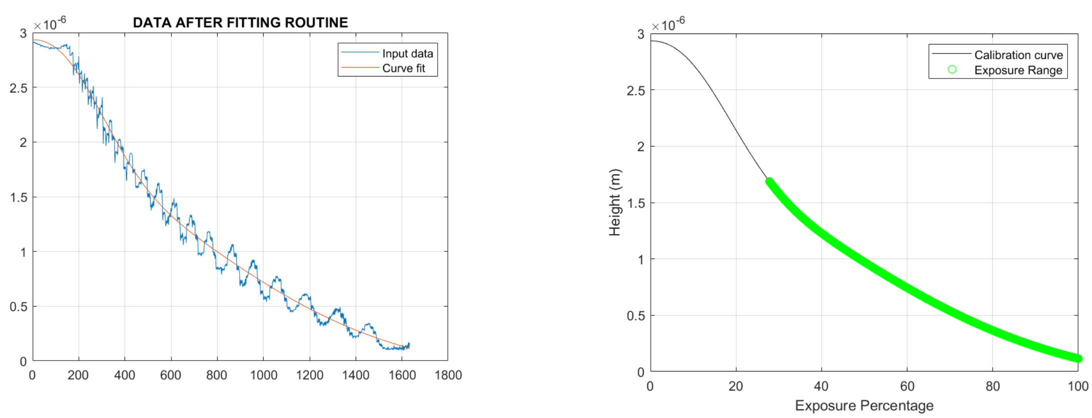


Figure 24. (left) Test ramp profile data fit to a curve. The oscillating raw data is due to the high reflectivity of the photoresist as well as small pixel size. (right) Exposure range in green corresponding to a 1.688 micron max-OPD etch depth.

Figure 24 (right) shows in green the exposure range that can be used to fabricate a CGH. Notice that the full exposure range is not used. This is because the full 0 to 255 grayscale range produces a max-OPD etch depth of approximately 2 microns. The ideal max-OPD etch depth to achieve for CGH fabrication is less than 2 microns, so the linearization software (provided in Appendix C) scales and maps the CGH grayscale values to a range that will produce the desired max-OPD etch depth. For example, to achieve a max-OPD etch depth of 1.688 microns, the grayscale range of 71 to 255 may be used. It is often desirable to include an offset, so that the CGH is fabricated using the middle range of exposure dosages. This requires the selection of an exposure range in which the photoresist's response to varied exposure dosages is relatively linear. Linearization for this project is done through printing ramps onto the intended substrate and using the measured surface profile data to determine the linear exposure range. The ramps printed at the corners with the CGH design facilitate easy verification of linearization after fabrication.

3.4 Grayscale Photolithography with the MLA 150

After linearization, the bitmap file is converted to a DXF file, which is a CAD file that specifies the physical dimensions of each feature in the design, as well as the grayscale value corresponding to each feature. This conversion script is described in Appendix C and provided in Appendix D. In the GUI for the MLA 150 maskless aligner, it is necessary to convert the DXF file into a job file that the machine can compile. Within the MLA 150, a 375 nm laser source passes through a 400x400 pixel digital micro-mirror array (DMD), which serves as a spatial light modulator, before being reflected towards a focusing lens [5]. This lens images pieces of the design exposure map down to the sample coated with photoresist. Given that the size of the

DMD is much smaller than the size of the desired write area, the stage on which the sample sits scans back and forth, allowing the writehead to expose the full write area.

3.5 Development or Wet Etching

After the exposure step, the sample is developed using a developer solution with 25% Microposit 351 developer and 75% de-ionized (DI) water. It is crucial that the sample be developed for the same amount of time as the test ramps fabricated for linearization are. This is to achieve the correct etch depth and maintain linearity. If one finds the need to change the development procedure, it is necessary to refabricate test ramps to obtain a linearization curve as the previous one will no longer be applicable.

3.6 Reactive Ion Etching with Plasma-Therm Versaline

Since the final fabricated CGH will be used with a high-power laser, it is necessary to use reactive ion etching to transfer the CGH pattern from photoresist into glass as photoresist cannot withstand prolonged exposure to high temperatures. The reactive ion etcher used for this project is the Versaline by Plasma-Therm.

To start, one needs to glue the 1-inch sample onto the 14-in sapphire carrier wafer. Wafers are transferred in and out of the etching chamber using a robotic arm or “handler,” which is why the use of a specially sized carrier wafer is necessary. The etching chamber is kept in a vacuum state with pressures on the order of a few millitorr. Prior to the etching phase, gases are pumped into the chamber in specified concentrations and stabilized. During the main etch phase, an RF bias is applied to the electrode on which the carrier wafer sits. This is called the “table bias” in the figure below and “HF bias” in the table below. Power is also applied to coils around the chamber to inductively couple energy into accelerated Argon ions, thus creating plasma [7]. The Argon ions experiences an oscillating electric field between the chamber itself, which is a grounded anode, and the table on which the carrier wafer sits, which is a cathode. Thus, the ions are accelerated vertically downwards towards the sample and remove photoresist [14]. Besides Argon, etching recipes may include other gases to stabilize the etching process, tune etch rates, and reduce noise in the final etched product. The development and tuning of reactive ion etching recipes is a field in and of itself that is beyond the scope of this thesis.

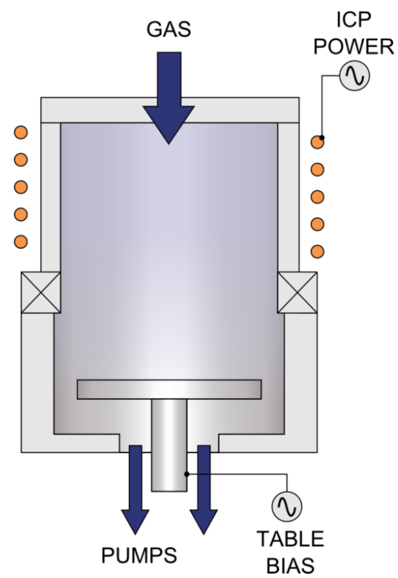


Figure 25. Illustration of reactive ion etching chamber courtesy of Oxford Instruments [7]

Reactive ion etching is commonly used to transfer binary patterns from photoresist to materials like silicon wafers, allowing for the precise fabrication of micro- and nanostructures. The act of transferring grayscale patterns from photoresist into glass using reactive ion etching is less common. For starters, the glass material must be specially chosen. Since fused silica is an undoped glass consisting purely of silica crystals or silicon dioxide (SiO2), an existing silicon dioxide etch recipe already programmed into the reactive ion etcher is used. The recipe was developed by IMTEK. It is necessary to decrease the ICP (inductively coupled plasma) power found in the original recipe to prevent the photoresist from overheating and vaporizing during the etch, which can cause an unfaithful pattern transfer. Details of the recipe are shown below with modifications highlighted in red. Note that only the "main etch" phase of the recipe is shown. The complete recipe consists of 6 steps, including ascending the wafer table or chuck to position, gas stabilization, and dechucking. More details regarding best practices for using the reactive ion etcher, as well as procedures for the optimal fabrication process are provided in Appendix A.

Selectivity (FS:PR)	FS Etch Rate (um/min)	PR Etch Rate (um/min)	ICP (W)	HF Bias (W)	Gas 1 C4F8 (sccm)	Gas 2 Ar (sccm)	Gas 3 O2 (sccm)	Pressure (mTorr)
1.4	0.56	0.4	1500	300	30	50	5.0	5
1.4	0.343	0.245	1000	300	30	50	5.0	5

Table 5. The row in grey is the original SiO2 etch recipe from IMTEK. Below that row is the modified recipe.

Note that decreasing the plasma power does not change the selectivity of the recipe, but it does reduce the etch rate. The selectivity is the ratio of the silicon dioxide etch rate to the photoresist etch rate. The selectivity is also equal to the scaling factor between the etch depth of a given feature in photoresist and that same feature when transferred into fused silica. Therefore, if the max-OPD etch depth of a CGH in fused silica needs to be 2.364 microns, the pattern needs to be 1.688 microns in photoresist to account for this differential etch scaling factor. See Appendix B for supplemental figures related to fused silica etching.

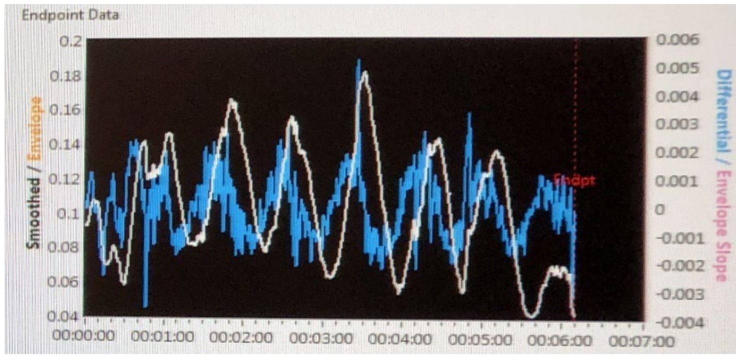


Figure 26. Interferometer data generated in manual endpoint window during main etch phase. The red dotted line shows when the operator manually stopped the etch.

There is a small 15-mm window at the top of the etching chamber allowing an interferometer to be trained onto part of the sample. During the main etch phase, the interferometer generates fringe patterns, and when the fringe patterns are interrupted, as shown in the figure above, the operator would know that they have etched through the photoresist into fused silica. However, relying on the interferometer to call the endpoint is not a reliable method to ensure that the correct etch depth is achieved as the fringe patterns are often noisy and difficult to interpret. As such, it is necessary to find the total etch time needed to etch through the photoresist. This is found by etching test ramps into a fused silica substrate and manually calling the endpoint using data from the interferometer. Since the depth of the test ramps is measured in photoresist before the RIE etch as well as afterwards in fused silica, by how much the sample is under-etched can be determined. This information allows for the calculation of the etch rate of the recipe, which is shown in table 5, as well as the total etch time.

Chapter 4

Results at 650 nm

4.1 Design 1

The first CGH design has an AU of 29 microns and a pixel count of 512x512. As noted in Chapter 2, a factor of 2 was incorrectly omitted when calculating the AU for this design, so while the intended angular extent θ_{max} of the hologram image is 2.56 degrees, and actual angular extent is 1.28 degrees.

The first design is calculated with the condition of a uniform or top-hat source, so a source distribution matrix is not applied in the Gerchberg-Saxton algorithm to find this design. Since the design is to be tested with a collimated laser diode, the source beam size is also not considered. After running over a thousand trials of the design algorithm, a CGH design producing a uniform array is found. The relative diffractive efficiencies of each of the +/- 10 diffraction orders are shown in the bar graph below. Recall that the absolute DE is the sum of the DE of the 20 signal orders.

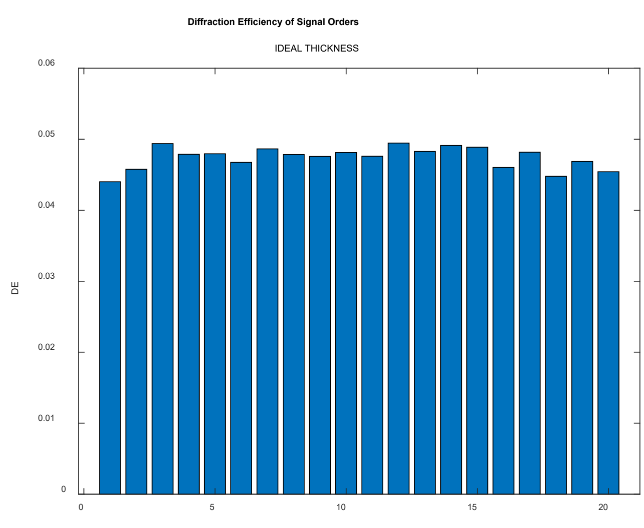


Figure 27. 650 nm Design 1 Simulated Diffraction Efficiency

Mean Relative DE	Standard Deviation of Relative DE	Standard Deviation / Mean	Uniformity	Absolute DE
4.71%	0.15%	3.25%	11.47%	94.3%

Table 6. Predicted Performance of 650 nm Design 1

650 nm Design 1 is fabricated and tested in photoresist only, and the ideal max-OPD etch depth is found to be 0.65/(1.62-1), which comes out to 1.048 microns. After deriving an etch depth vs. exposure dosage curve from test ramp data, an exposure range is selected to be able to achieve the desired etch depth. For fabrication, a 64-bit grayscale table in the MLA 150 is used, which meant that after linearizing the CGH design, grayscale values had to be mapped from the 0 to 255 range to the 0 to 63 range. A grayscale table is a table listing the exposure dosage that corresponds to each grayscale value.

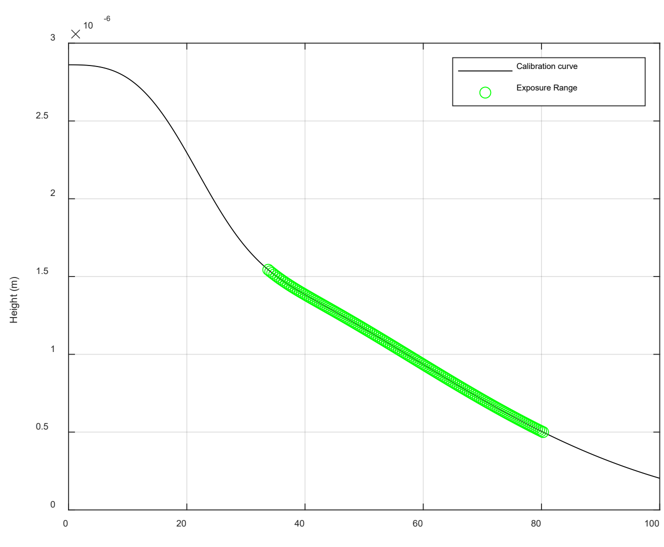


Figure 28. Exposure range in green for 650 nm Design 1

This design is fabricated twice (CGH 1 and 2 in the table below) in the same way to determine reproducibility. The table shows that etch depths are close to the +/- 5% etch depth tolerance. Etch depths are measured using the NT9800 optical surface profiler by Veeco.

Ramp #	CGH 1 Etch Depth	CGH 1 Depth Error (%)	CGH 2 Etch Depth	CGH 2 Depth Error (%)
1	0.9689 um	-8.2%	1.1268 um	6.8%
2	1.1298 um	7.1%	1.0601 um	0.5%
3	1.0845 um	2.8%	1.0388 um	-1.6%
4	1.0306 um	-2.3%	1.0870 um	3.0%

Table 7. “Etch Depth” here refers to the depth corresponding to the maximum OPD

After determining that the design had been fabricated satisfactorily, a simple testing setup is constructed. The CGH is illuminated with a collimated red laser diode source, with an iris diaphragm used to adjust the beam size to the side length of the CGH active area. Since the diffraction angles introduced by the CGH is less than 1 degree, it is necessary to use a focusing lens to shorten the physical path to the image plane, allowing it to be measured by the RGB camera (Thorlabs CS135CU).

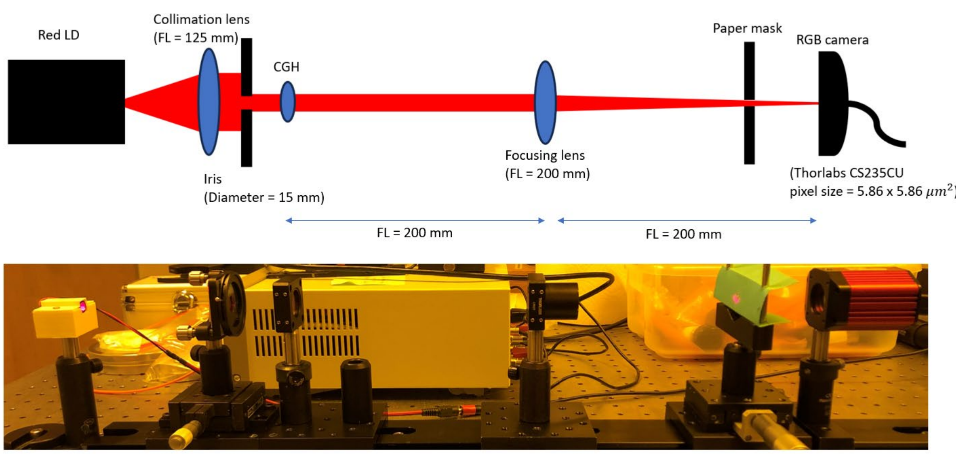


Figure 29. 650 nm Initial Testing Setup

Upon illuminating the CGH with the collimated laser diode, it is found that many higher order images of the target are produced. To isolate the 0^{th} order image, a paper mask is used to block higher order copies. The figure below to the right shows a square hole in the paper mask letting the 0^{th} order image pass through. Higher order copies lie on either side of the hole, made visible by the paper.

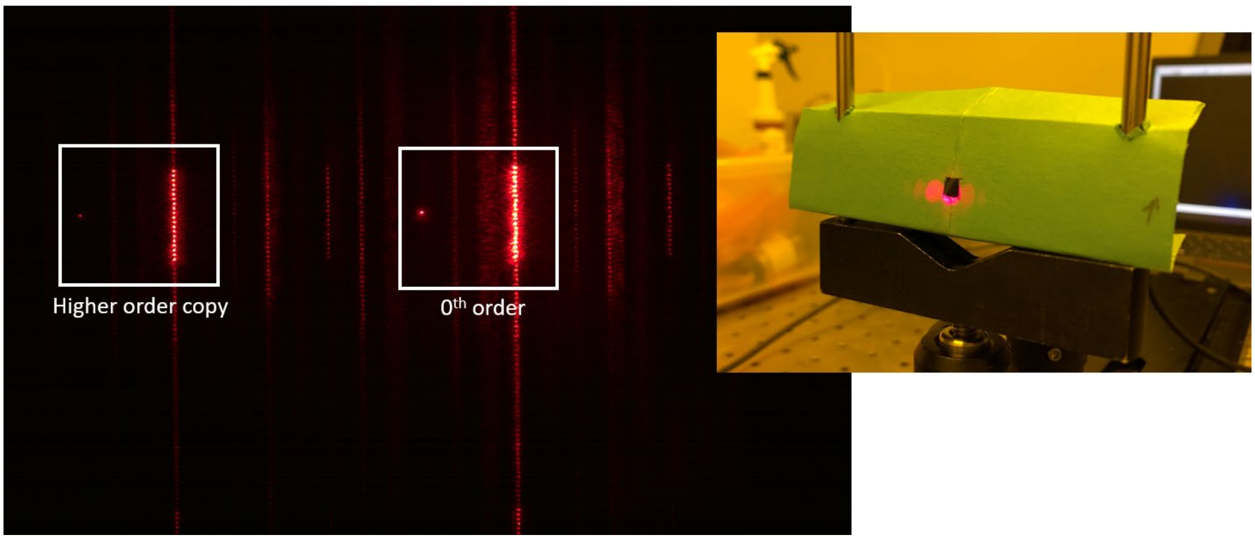


Figure 30. (left) Raw image showing 0^{th} order and higher order image. (right) Paper mask used to block higher order images.

With knowledge of the focal length of the focusing lens and the pixel size of the RGB camera, the experimental diffraction angle is derived from a raw image. The separation between signal orders is first found in units of pixels, shown as Δx in the equation below, before being converted to degrees.

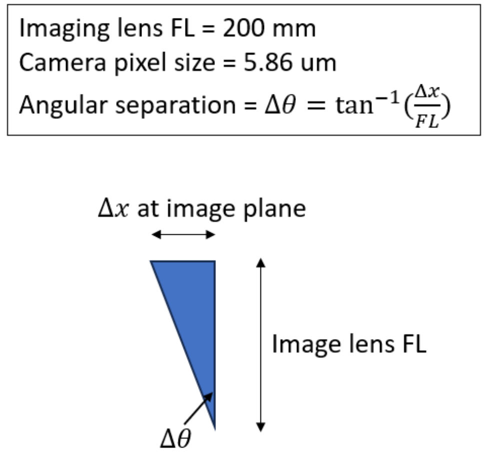


Figure 31. Schematic and equation showing how diffraction angles are calculated.

It is found that all diffraction angles are exactly half of what is intended.

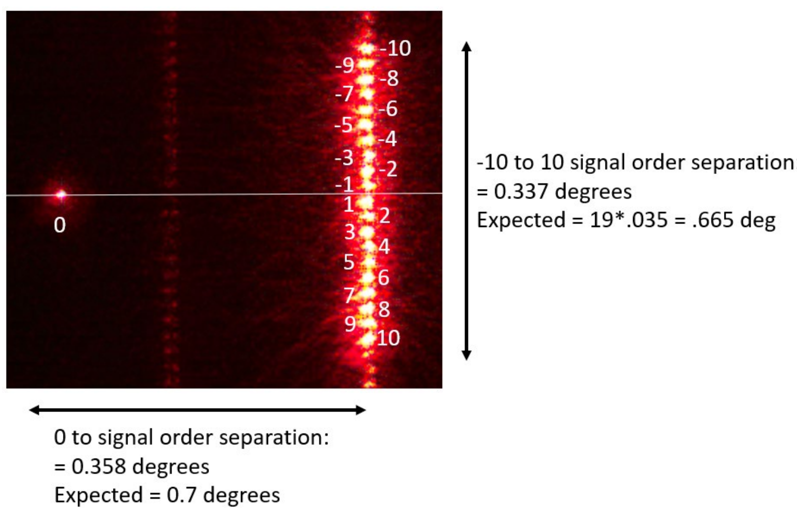


Figure 32. 650 nm Design 1 diffraction angles

Absolute diffraction efficiency is also estimated using a power meter (Thorlabs PM320E). First, the power before and after the CGH are measured. Then, a slit is used to isolate the 0^{th} and +/- 10 orders so that their power could be measured. The below calculations show that the percentage of power going into to the signal orders is only 26.5%, which is well under the specification. On the other hand, the 0^{th} order DE is minimized, meaning the CGH is fabricated to the correct etch depth.

Before CGH	234.3 uW
After CGH	213.9 uW
After Mask (0^{th} order)	2.1 uW
After Mask (+/- 10 orders)	56.7 uW

$$\text{Signal orders diffraction efficiency} = \frac{P_{after\ mask\ (\pm 10 orders)}}{P_{after\ CGH}} = \frac{56.7}{213.9} = 26.5\%$$

$$0^{th}\ \text{order diffraction efficiency} = \frac{P_{after\ mask\ (0th\ orders)}}{P_{after\ CGH}} = \frac{2.1}{213.9} = 1\%$$

Table 8. 650 nm Design 1 diffraction efficiency.

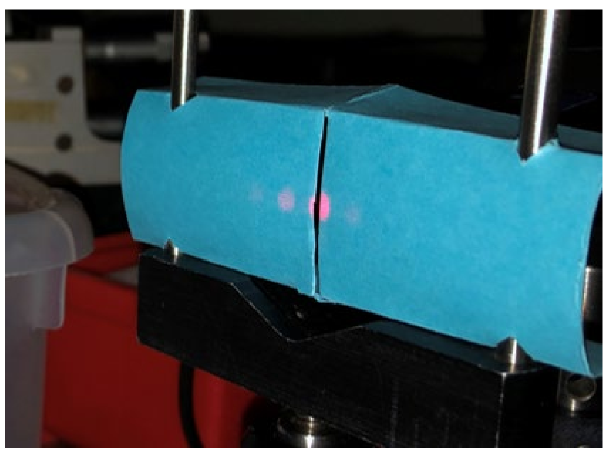


Figure 33. Slit used to isolate the 0^{th} and signal orders.

An estimation of beam uniformity is also performed using the raw image. This is done through cropping the image around the area of the signal orders, applying a gaussian mask on each of the signal orders, and finding the sum of pixel values within each mask. Uniformity is found to be approximately 20%. Since the camera is saturated by the signal order beams, it is determined that this uniformity estimate may not be accurate.

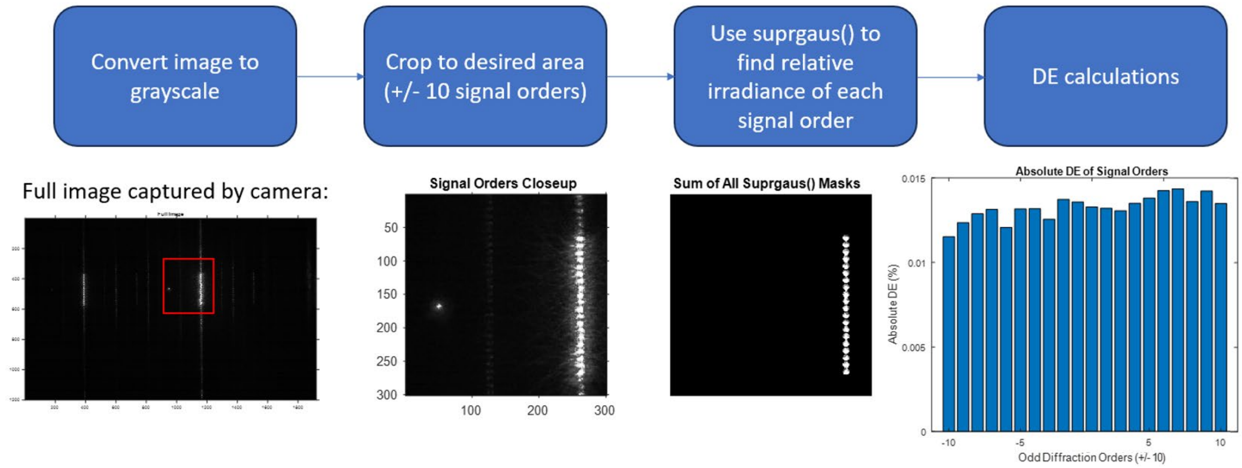


Figure 34. 650 nm Design 1 uniformity estimation process and result.

One can notice that every aspect of this first design is to simplify matters as much as possible to obtain a first proof-of-concept demonstration. The source size and source distribution are not considered; the design is fabricated in photoresist; the 64-bit grayscale table is used instead of the 256-bit grayscale table. Two main takeaways from this first design are first, that a factor of 2 is missing from the equation linking the angular extent of the image and the AU and second, that a source distribution needed to be used in the design algorithm to improve the absolute diffraction efficiency.

Besides the presence of higher order false images, there is a high degree of noise in the image. Simulations done in MATLAB showed that non-linearity is the cause of the noise. A nonlinear curve is pulled from previous test ramp data to find the simulated image of a non-linearly fabricated CGH. To see the noise level in the simulation, all pixel values above a threshold are set to the same value, which simulates a saturated camera detector. A side-by-side comparison shows that the simulated image correlates well with the raw image captured by the RGB camera.

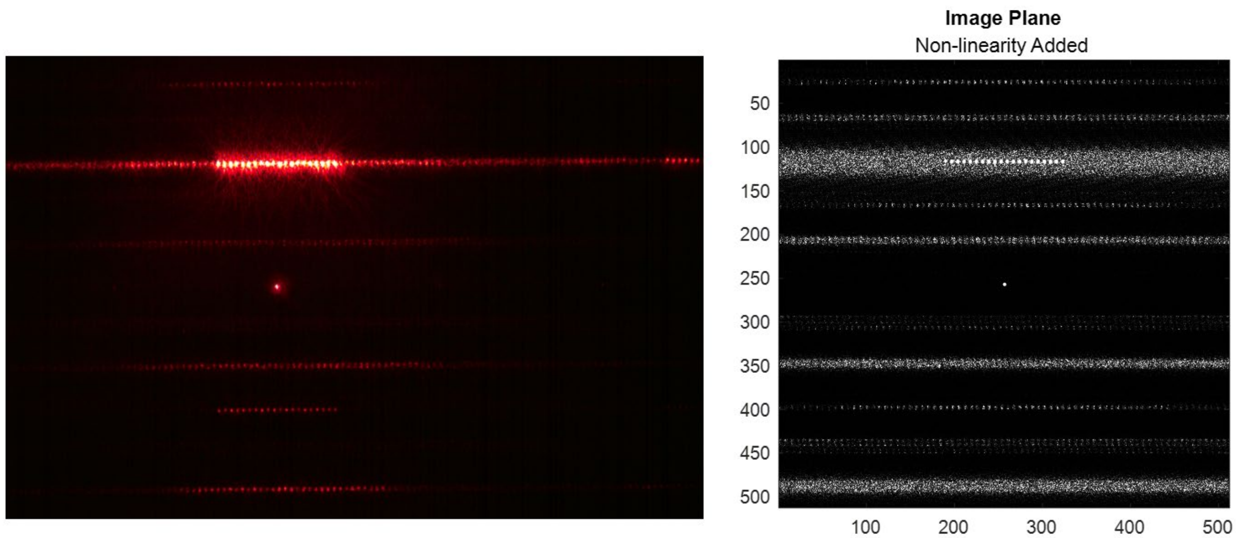


Figure 35. (left) Raw image from 650 nm Design 1. (right) Simulated image of non-linear CGH.

4.2 Design 2

For Design 2, the AU is changed to 15 microns to correct the diffraction angle found in Design 1. An image of the source at the CGH plane is captured using the RGB camera, and this image is converted into a source matrix that is applied in the design algorithm. Once again, the design algorithm is run multiple times to search for the design with the most optimal uniformity. In terms of fabrication, the etch depths are once again the +/-5 tolerance, meaning that the exposure and development procedure being used is producing consistent results and does not need to be adjusted. For Design 2, the diffraction angles are found to be correct and within the 5% tolerance.

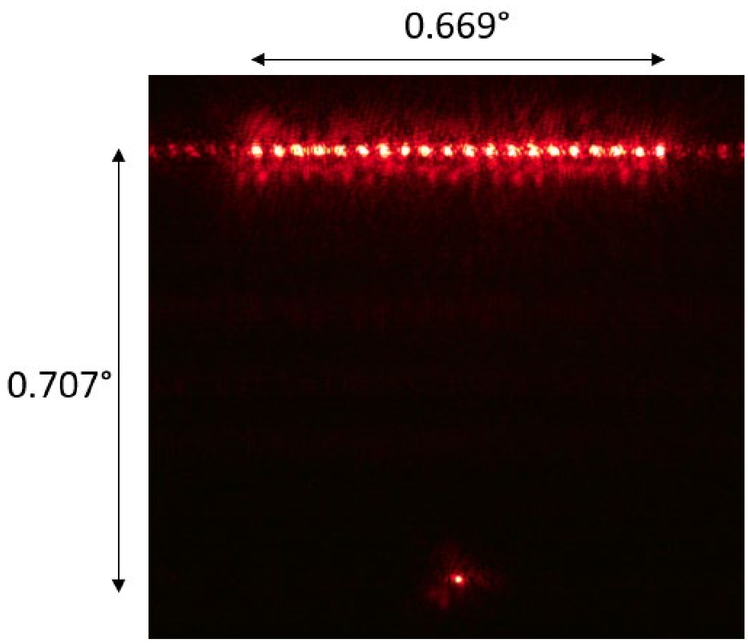


Figure 36. 650 nm Design 2 diffraction angles.

There is also a significant reduction in noise level after a new linearization curve is found through refabricating test ramps with more levels. The graph below shows the surface profile data of a test ramp placed at one of the corners of the fabricated CGH. The linearity of the test ramp indicates that the rest of the CGH is linearized properly.

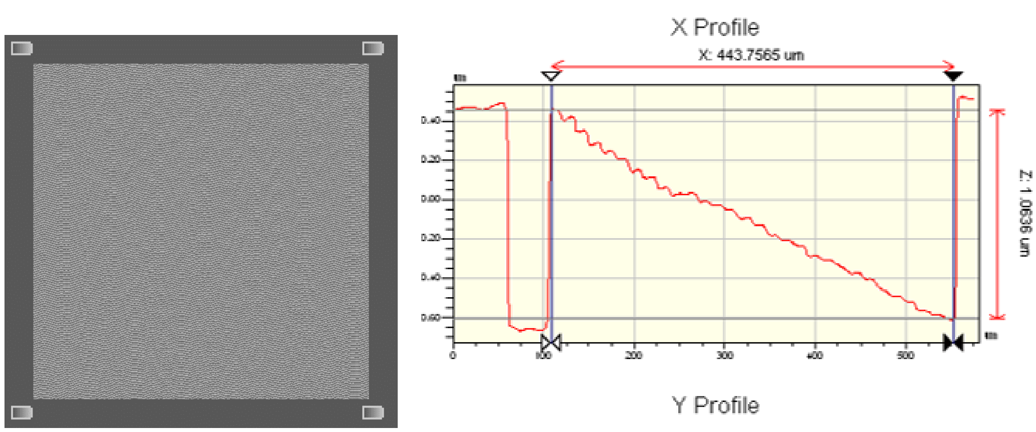


Figure 37. (right) Design 2 linearized, with test ramps placed in each order. (right) Surface profile data from one of the ramps.

Like with Design 1, diffraction efficiency is measured using a power meter and translatable slit. The absolute diffraction efficiency of Design 2 is 68%, which is significantly improved compared to 26% in Design 1.

	Power	Image Label #
Before CGH	0.625 uW	
After CGH	0.552 uW	
0th order	0.015 uW	1
+/- 10 Signal Orders	0.374 uW	2
Higher order image	0.087 uW	3

$$DE_{signal\ orders} = \frac{P_{signal\ orders}}{P_{after\ CGH}} = \frac{0.374}{0.552} = 68\%$$

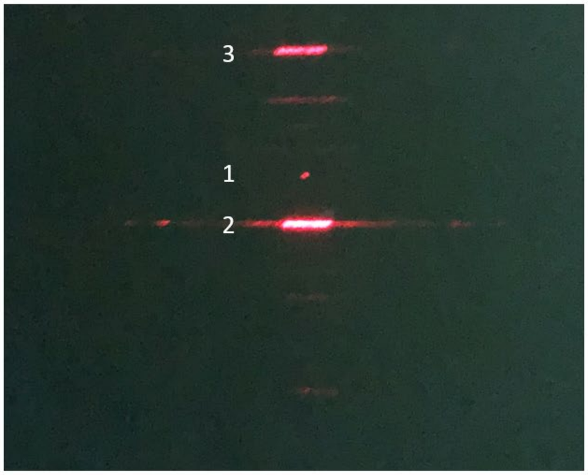


Figure 38. 650 nm Design 2 diffraction efficiency measurement.

It is observed that the camera is still saturated, so a 40X microscope objective is used to image the signal orders individually. It is found that the shape of most beams in the array are mishappen. This is because the design algorithm had been sampling the image bitmap fed into the algorithm and finding a CGH that would reconstruct the sampled image. For this project, the dimensions of the CGH and image bitmap are kept the same, so the sampling portion of the design algorithm is removed for Design 3.

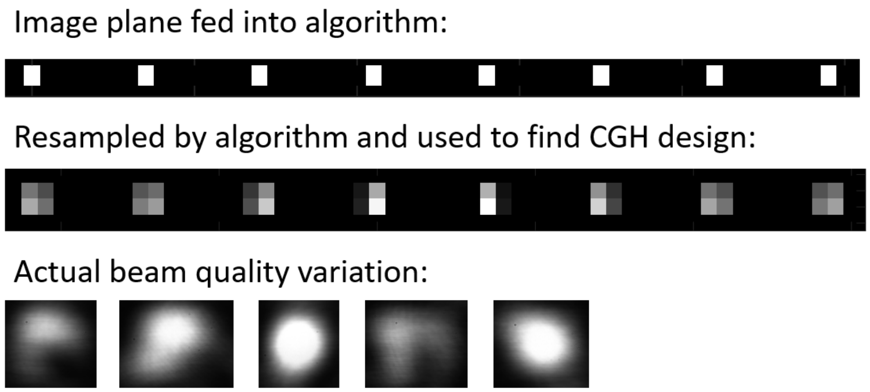


Figure 39. Explanation of why beam shapes varied across the array for Design 2.

4.3 Design 3

For Design 3, the AU remains at 15 microns while the pixel count is increased to 1024x1024. It is found that increasing the pixel count allowed the design algorithm to find a more uniform design. However, due to the source distribution, the design algorithm is unable to find a design that meets the uniformity specification of 10%. The predicted uniformity for the design is 28%, and the estimated actual uniformity of the fabricated CGH is 51%. This same ratio of 2 between the actual uniformity and predicted uniformity is also observed in Design 1.

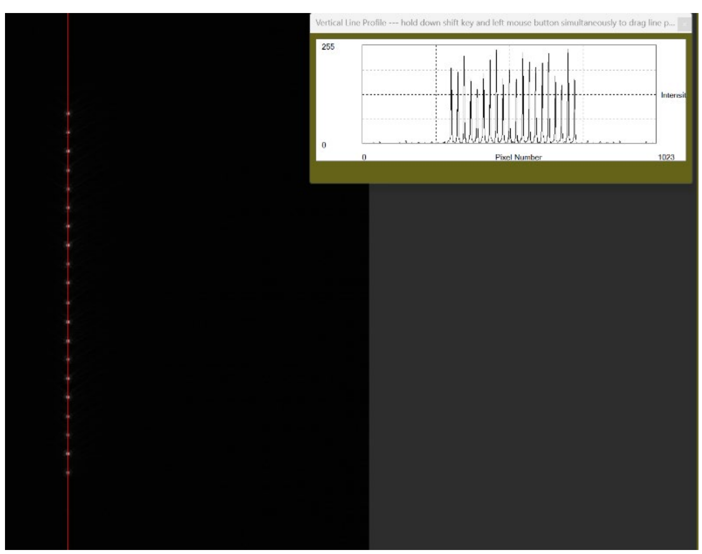


Figure 40. Irradiance profile in Thorcams of 650 nm Design 3.

With the design algorithm no longer sampling the target image file, beam shape is much improved in Design 3 compared to Design 2.

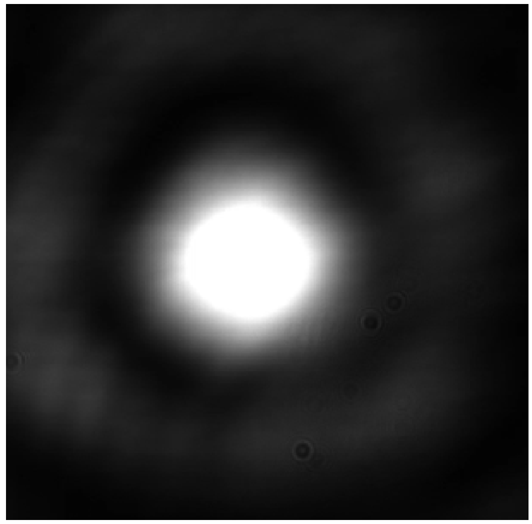


Figure 41. Closeup image of a signal order produced by 40X microscope objective.

There is also a significant improvement in diffraction efficiency, likely because this design is fabricated using the 256-bit grayscale table. It is shown in Chapter 2 that the increase of quantization levels reduces false images, which increases the diffraction efficiency. The absolute diffraction efficiency is found to be 85%, up from 68% from Design 2. The 0^{th} order DE is 2%, which is well under the 5% tolerance.

	Power (uW)	% of After CGH Power
Before CGH	0.927	
After CGH	0.884	
0th order	0.018	2.04%
Signal Orders	0.755	85.41%
2nd order image right	0.048	
2nd order image left	0.025	
SUM	0.846	

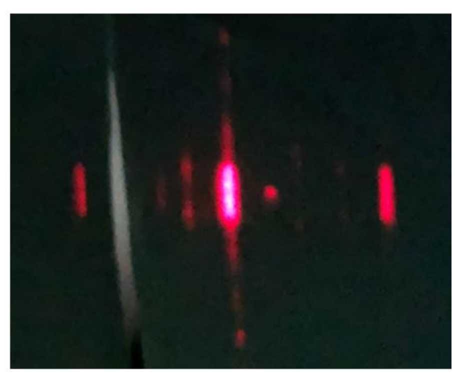


Figure 42. 650 nm Design 3 diffraction efficiency measurement.

4.4 Summary

Taken together, the 650 nm phase of the project was extremely valuable in terms of encountering and addressing unforeseen design and fabrication issues. Testing results from each design yielded valuable lessons that are applied to improve the next design. The most important results from this phase of the project were the correction of the AU equation, the importance of considering the source distribution, and the importance of linearization.

Chapter 5

Results at 1064 nm

5.1 Design 1

The first design for 1064 nm has an AU of 24 microns and a pixel count of 512x512, so the fabricated CGH is 12.3x12.3 mm. The source distribution matrix used for the design accounts for the ratio between the beam diameter of 13.4 mm, and the CGH side length of 12.3 mm. Since the test source is Gaussian, there is a large difference between the irradiance at the center of the CGH area and the edge. This makes it difficult for the design algorithm to find a CGH design that would produce a uniform beam array given that the source is non-uniform. To get a first off-site testing result with the 1064 nm source, an initial design is chosen and fabricated.

This first design is fabricated in photoresist as process development for the fused silica etch is still underway. Since the source distribution matrix is well-tailored to the test source, there is a high degree of correlation between test result and the predicted result. The predicted uniformity of 38% matches the actual uniformity to within a percent.

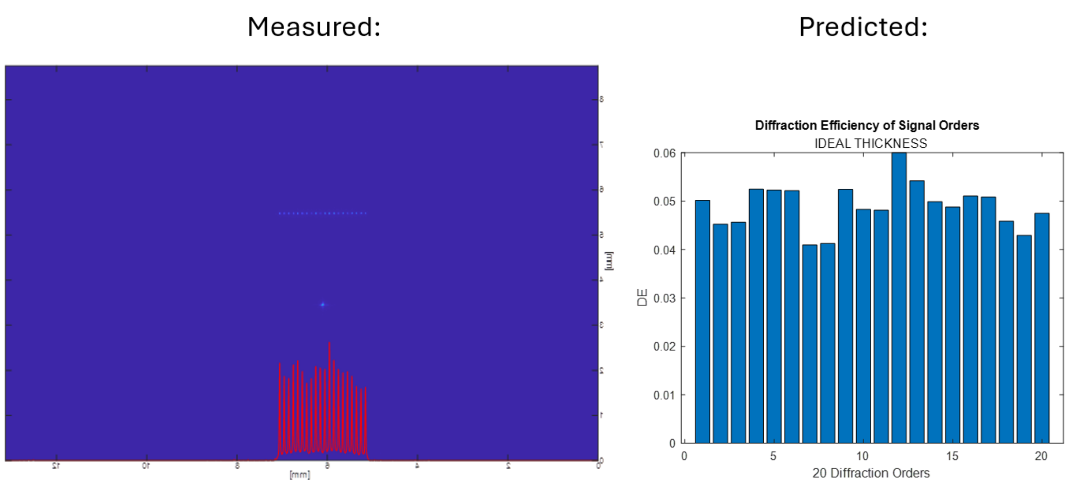


Figure 43. (left) Raw image with red irradiance profile overlaid in red. Image courtesy of NRCA. (right) Predicted beam array diffraction efficiency.

5.2 Design 2

The CGH size is increased for Design 2 to capture more of the beam profile, so the AU is changed to 14 microns and the pixel count to 1000x1000. The target image and source distribution are redesigned to account for the change. To find the design, the ROI/RONI modification is added to the design algorithm. This yielded a design with reduced absolution diffraction efficiency, but uniformity that is close to the 10% specification.

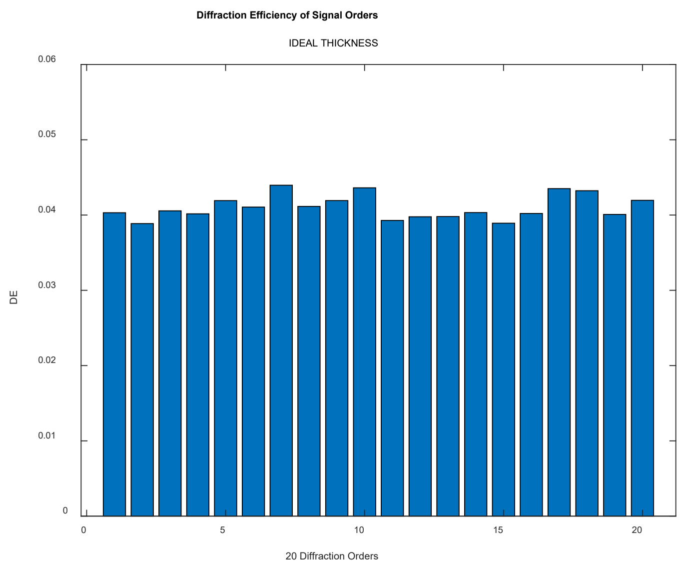


Figure 44. Initial predicted signal order diffraction efficiencies for 1064 nm Design 2.

Mean Relative DE	Standard Deviation of Relative DE	Standard Deviation / Mean	Uniformity	Absolute DE
4.1%	0.16%	3.86%	12.42%	82%

Table 9. Initial Predicted Performance of 1064 nm Design 2

This design is fabricated both in photoresist and in fused silica. Experimentally, this allows one to distinguish between what is due to the design itself, which can be done through measuring the photoresist sample, and what is due to the transfer of the design to fused silica. For the photoresist sample, the max-OPD etch depth is 1.064/(1.61-1) = 1.72 microns. Note that the index of the photoresist is 1.61 for 1064 nm, compared to 1.62 for 650 nm. For the fused silica, the pattern is first etched in photoresist with a max-OPD depth of 1.68 microns. The etch depth is scaled up by 1.4 due to the selectivity of the silicon dioxide etching recipe used, so the max-OPD etch depth in fused silica comes out to be 1.4*1.68 = 2.37 microns =1.064/(1.45-1). In this way, the correct etch depth in fused silica is achieved by taking into account the scaling factor introduced when transferring the pattern from the photoresist into fused silica.

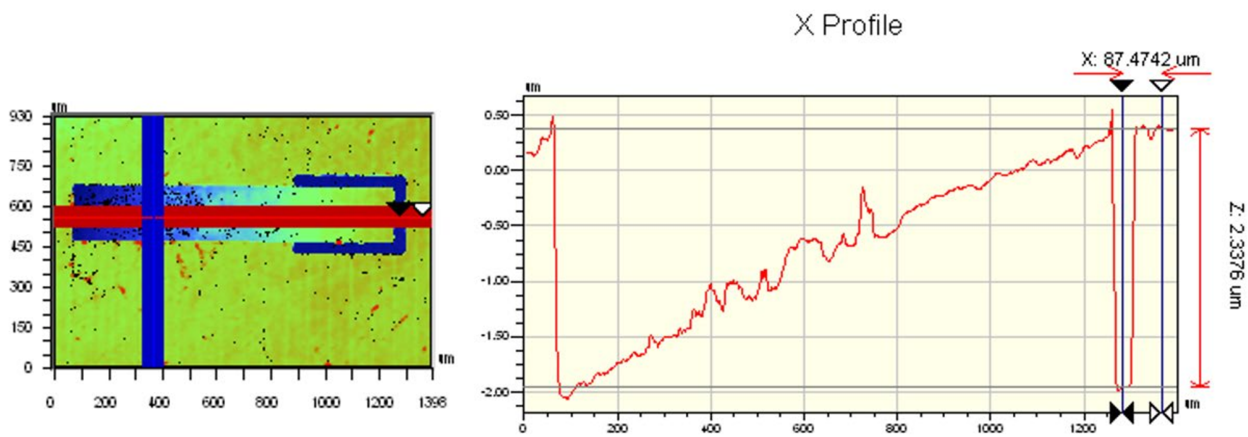


Figure 45. (left) Phase map of test ramp etched in a corner along with the CGH design. Median filter applied. (right) The max-OPD etch depth is shown to be 2.34 microns, which is within the +/- 5 etch depth tolerance.

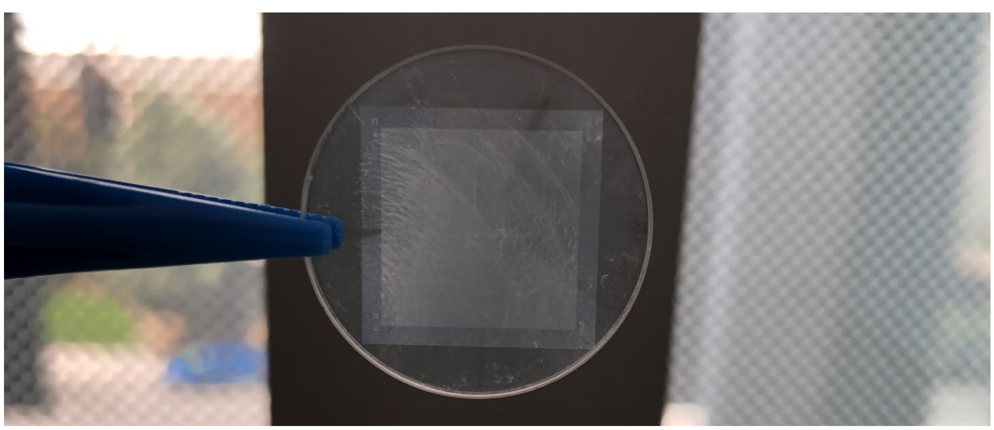


Figure 46. Photo of 1064 nm Design 2 etched in fused silica.

For 1064 nm design 2, there is once again a high correlation between the simulated result and the experimental result. However, the measured uniformity for both the fused silica and photoresist samples are about two times the predicted 12%. This discrepancy is due to not taking the square root of the source distribution matrix before applying it within the design code. The source distribution matrix is a constraint on irradiance, and the Gerchberg-Saxton algorithm works with field amplitude. The updated simulated result is shown below.

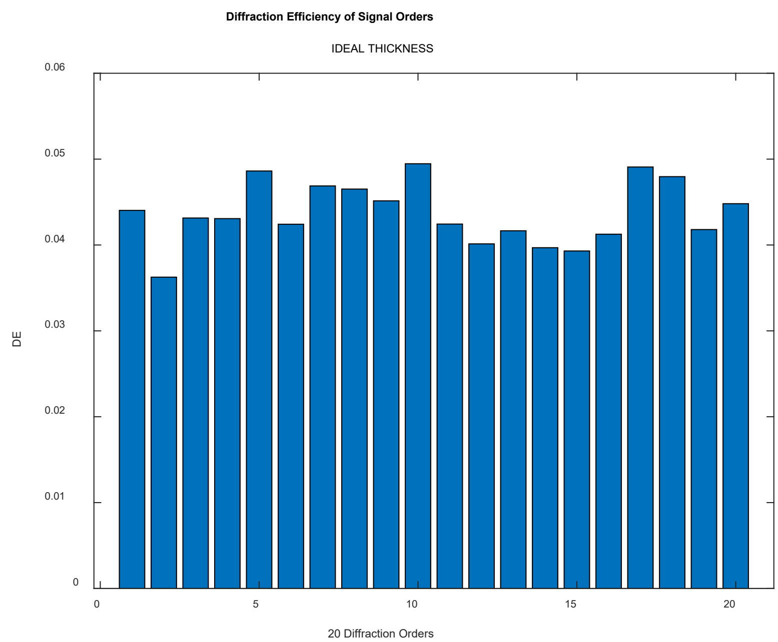


Figure 47. Corrected prediction of signal order diffraction efficiencies for 1064 nm Design 2.

Mean Relative DE	**Standard Deviation of Relative DE**	**Standard Deviation / Mean**	**Uniformity**	**Absolute DE**
4.4%	0.36%	8.26%	30.23%	87.36%

Table 10. Corrected Prediction of Performance of 1064 nm Design 2

The 0^{th} order diffraction efficiency is well within the 5% tolerance for the photoresist sample, but this is not the case for the fused silica sample, likely because the max-OPD etch depth is not uniform across the entire CGH area. During the process development for the fused silica etch, it is found that temperature played a large role in determining the etch rate. During the etching process, the temperature across the area of the photoresist-coated sample could have been uneven, causing slight differences in etch depth. Another contributing factor is the high degree of noise or randomly scattered pits present in the fused silica after the etch. These pits and blemishes are due to the photoresist overheating and slightly vaporizing during the etch.

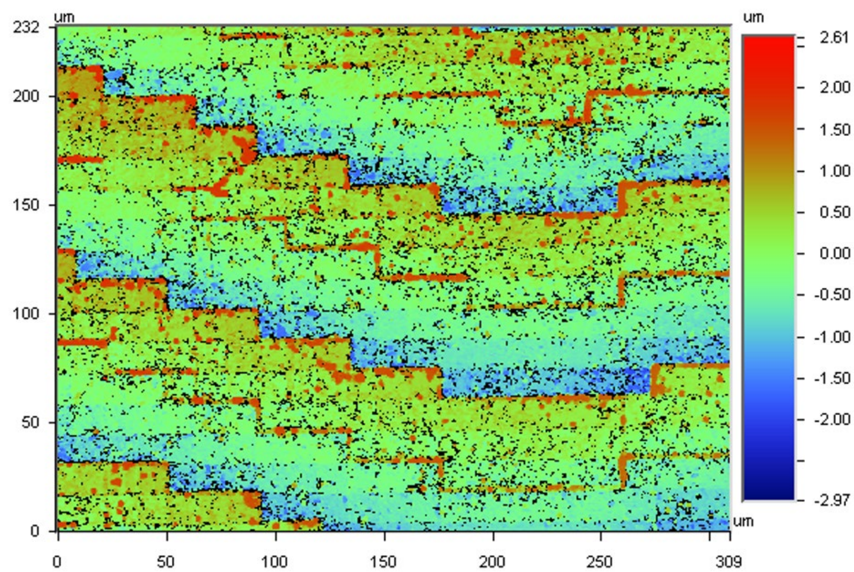


Figure 48. Phase map generated by the NT9800 showing a high degree of noise in a CGH design etched into fused silica

5.3 Design 3

A third design is found with the corrected application of the source distribution in the design software. As shown in Table 11, the predicted uniformity and diffraction efficiency of Design 3 is approaching the specifications from Table 2.

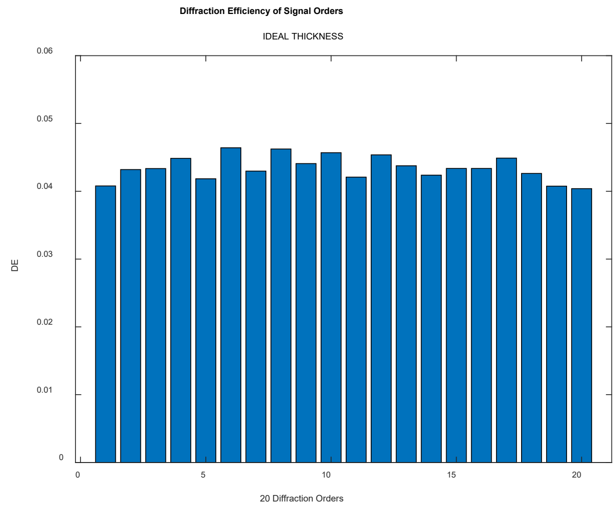


Figure 49. Prediction of signal order diffraction efficiencies for 1064 nm Design 3.

Mean Relative DE	**Standard Deviation of Relative DE**	**Standard Deviation / Mean**	**Uniformity**	**Absolute DE**
4.34%	0.18%	4.09%	13.88%	87%

Table 11. Prediction of Performance of 1064 nm Design 3

After many process development experiments, it is determined that the way to eliminate the presence of noise structures previously observed is to bake developed photoresist at 115°C for an hour. This addition to the fabrication process was triggered by the study of previous work with the S1827 photoresist done at the University of Louisville [8].

Two samples of Design 3 are fabricated for testing, one on a 1mm thick substrate, and one on a 2 mm thick substrate. The fabrication procedures used are provided in Appendix A. The fabrication results are below, captured using the Veeco Wyko NT9800 Optical Surface Profiler. At the time of writing, these samples have yet to be tested, but a significant improvement from Design 2 in terms of both uniformity and diffraction efficiency is predicted.

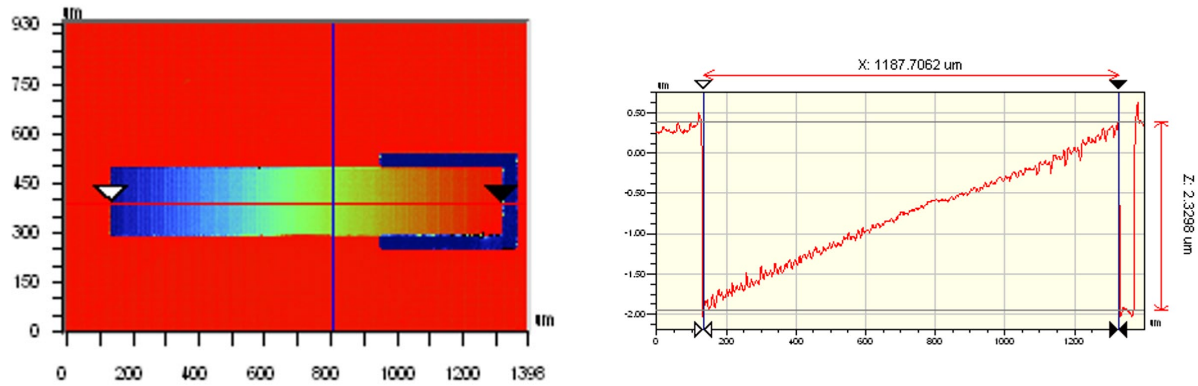


Figure 50. (left) A test ramp on the 1 mm sample. (right) Etch depth data for the test ramp.

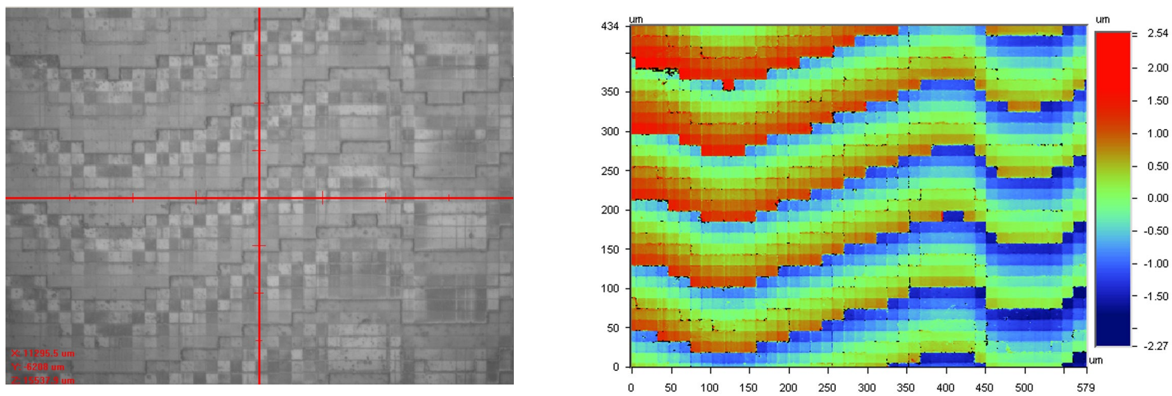


Figure 51. (left) Photo of CGH area for 1 mm sample. (right) Map of measured etch depths.

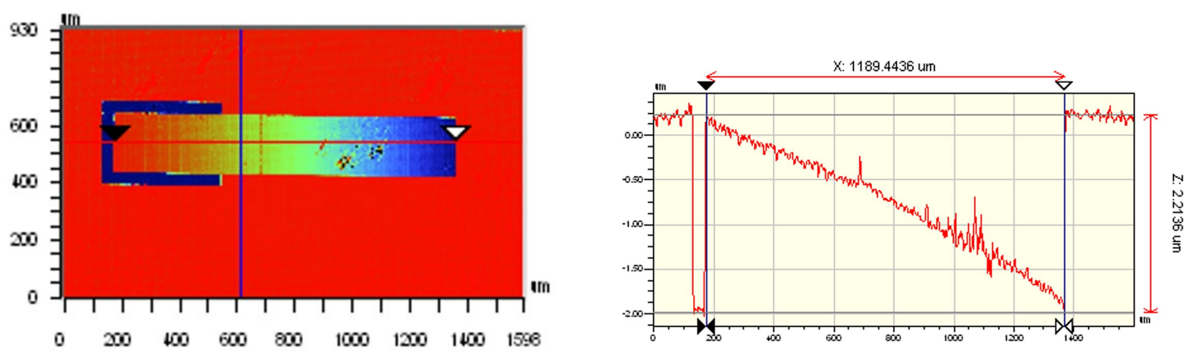


Figure 52. (left) A test ramp on the 2 mm sample. (right) Etch depth data for the test ramp.

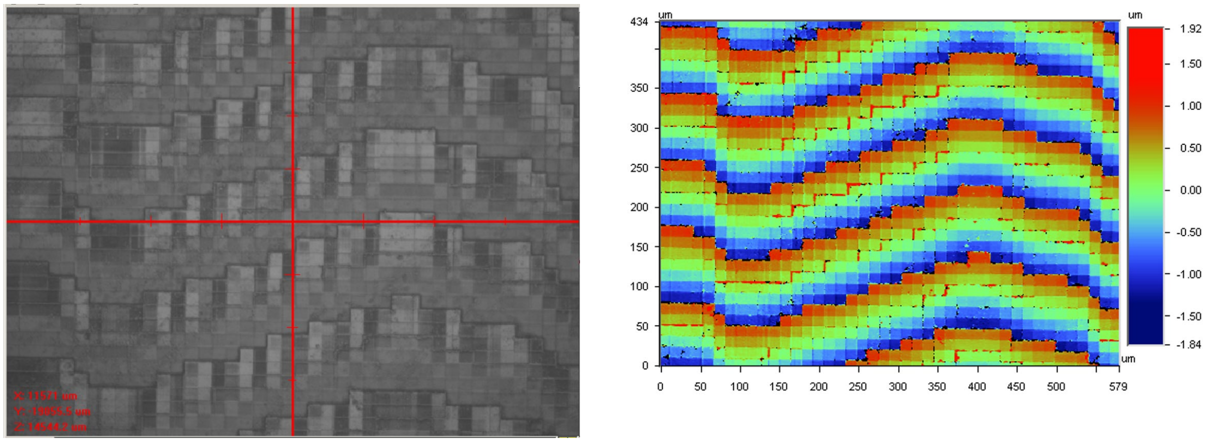


Figure 53. (left) Photo of CGH area for 2 mm sample. (right) Map of etch depths measured.

The 1 mm sample is slightly cleaner than the 2 mm sample in terms of unwanted noise structures, but both show a significant improvement when compared to the fused silica sample for Design 2. A thinner substrate more effectively transfers heat from the surface being etched down through the carrier wafer to the electrode table being held at a cooler temperature. Note that typical RIE processes involve wafers less than a millimeter thick, so it should not have been surprising to find that the 2 mm substrates used in this project presented a considerable challenge when it came to getting a clean, controlled etch.

Chapter 6
Applications and Future Work

6.1 Applications

Diffractive optical elements (DOE) capitalize on the wave description of light, allowing light to be spread out angularly with a high degree of precision (to less than 0.005 degrees for this project). They can be useful for a variety of applications, such as in optical metrology. A grating spectrometer, for instance, takes advantage of the fact that a holographic grating will diffract different wavelengths of light by different angles, thus allowing a camera to capture the spectrum of a given light source [6]. In the commercial product sector, diffractive optics are often used in the never-ending quest to make optical systems more compact. In virtual reality headsets, for instance, transmission gratings are often used to couple light into light pipes.

Dammann-style gratings, in particular, provide a compact way of generating a grid of uniformity spaced laser beams of equal irradiance. This function can be useful in a variety of optical systems to replace reflection-based beam-splitting setups, which are often less compact. Laser beam arrays can be used in interferometry, metrology, and optical computing.

Being able to create a grid of uniformly spaced and equal-irradiance laser beams is of particular use for 3D imaging. The uniform grid of laser beams generated by a Dammann grating is reflected off a 3D object. Certain laser beams in the grid will deviate from their original relative position upon reflection and relative irradiance between beams may change due to absorption properties of the object. With this information, it is possible to perform a 3D reconstruction of the object. This application has already been implemented commercially. Apple's Face ID [11] function projects a pattern of more than 30,000 infrared beams onto a user's face to create a 3D reconstruction of it, which can then be used for identification purposes.

6.2 Future Work

Further process development is needed to find a fused silica etch recipe that prevents unevenness in etch rate across the sample and the clouding of 1 mm substrates. Experts in plasma etch recipe process development may be consulted to troubleshoot issues encountered in this project.

On the design front, further work is needed to determine whether the diffraction efficiency and uniformity achieved in this project are approaching the limits of what is possible with a Dammann-style grating, or whether other design search and optimization techniques may be used to further improve beam generation performance. A promising direction is the implementation of a blocked simulation annealing algorithm in which a CGH design is decomposed into equally sized blocks and individually optimized. An objective function is called during the optimization of each block, and it works by inserting a given block with perturbations from simulated annealing back into the full original CGH design to evaluate whether the perturbations should be retained. Figure 54 below shows that blocked simulated annealing can improve uniformity while minimizing loss in total diffraction efficiency.

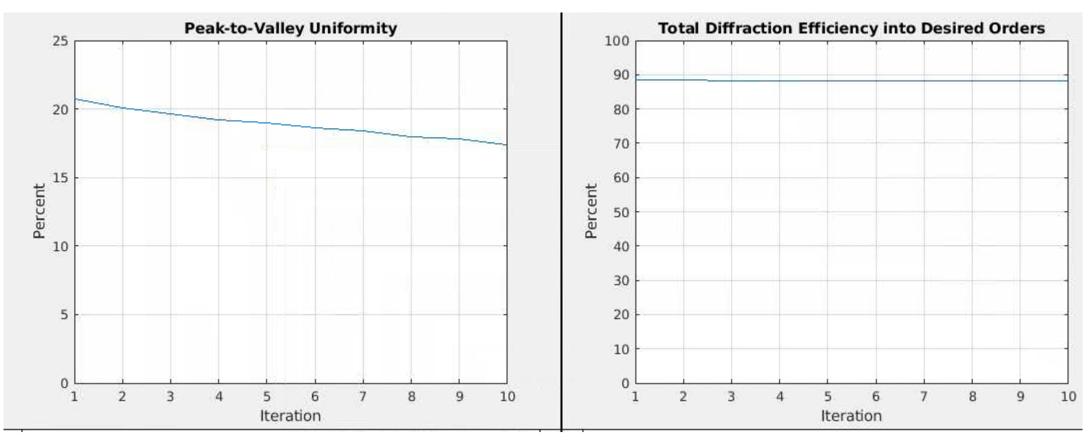


Figure 54. Figure courtesy of Tom Milster showing the cumulative result of running a blocked simulated annealing algorithm for 60 hours on a 94-core high performance computer.

Chapter 7

Conclusion

This DOE project started with an initial phase design iteration phase at 650 nm, which allowed for the timely correction of unforeseen design and fabrication issues as well as the learning of valuable lessons. In design 1, it is discovered that there is a missing factor of 2 in the equation relating the angular extent of the reconstructed image and the pixel size or AU. Discrepancy between the measured beam array diffraction efficiency and the predicted efficiency prompted the introduction of the source distribution matrix for design 2. Unexpected noise in the image plane is found to be caused by non-linearity. For design 2, noise levels in the image are reduced through re-obtaining linearization data, and the diffraction efficiency is much improved because of the implementation of a source distribution matrix in the design algorithm. However, it is found that there is a sampling section in the code that caused all the beams to be misshaped. This is corrected in design 3, along with improved diffraction efficiency.

At 1064 nm, there is high cohesion between simulated and measured results, especially for the relative diffraction efficiency of signal orders. This is likely because the source distributions used for this phase of the project are well tailored to the specified test source, and because a more optimal off-site testing setup is used. The measured uniformity fell short of the 10% specification for all three samples tested, so further work is needed to both search for better designs as well as make the fabrication process more controlled and consistent.

Fused silica etch process development is done in parallel with designs 2 and 3 at 650 nm as well as design 1 at 1064 nm. An appropriate silicon dioxide recipe is selected, and multiple test ramps are printed on photoresist and etched into fused silica to determine the etch rate of the recipe for various ICP power setpoints. Even though much work needs to be done in terms of recipe adjustment to prevent uneven etching into fused silica and the clouding of thinner substrates, the information that has been gathered up to this point allowed for a fused silica sample to be fabricated and tested. Two more fused silica samples have been fabricated and will be tested. These samples were fabricated using the optimal fabrication procedures detailed in Appendix A. Besides the high 0^{th} order DE caused by not achieving the ideal max-OPD etch depth, the results are promising and are highly correlated with simulated results.

The work presented in this thesis will be presented in a poster session in the "14^{th} International Conference on Optics-Photonics Design and Fabrication" (ODF'24, Tucson).

Appendix A

Summary of Optimal Fabrication Process

The following subsections detail the procedures for the most effective fabrication process found. A summary table of all machine settings is included at the end.

A.1 Substrate Prep

1. Turn on the hot plate and set its temperature to 120°C.
2. Check the settings for the "1: Static" process in the spin-coater. It should be 3000 rpm for 30 seconds. The acceleration is 1000 rotations/min/second, so after 3 seconds it reaches the target speed. With these spin-coater settings, the photoresist layer will be approximately 3.25 microns [10].
3. Attach a 0.2 micron Sartorium PTFE filter to a 2 ml Henke-Ject syringe. Remove the plunger from the syringe and fill the syringe about three-quarters full with S1827 photoresist. Reattach the plunger.
4. Rinse a substrate with acetone, then isopropanol, then DI water. Use the nitrogen gun on the wet bench to dry off the substrate. It is advised to apply the different chemicals to the substrate in quick subcession to prevent leaving stains.
5. Place the cleaned substrate in the spin-coater, turn on the vacuum pump, close the lid of the spin-coater, and start the "1:Static" process.
6. Once the spin-coater has finished, turn off the vacuum pump, remove the sample, and check for any blemishes or signs of unevenness on the photoresist. If the photoresist is deemed unacceptable, return to step 3 and proceed from there.
7. Place the sample on the hot plate and let it bake for 90 seconds.
8. Let the sample cool down for at least 3 minutes before placing it on the stage of the MLA 150.

A.2 Exposure

1. Copy a prepared DXF file into the folder named "dxf" on the MLA 150 computer.
2. On the MLA 150 interface, select "new job." Click the box below "Substrate Template" and select "1 inch round." Click the box below "Design" and in the window that pops up select "convert design." The converter window should pop up.
3. In the converter window, click "File," "New Job," and press enter. In the drop-down menu under "Source File," select "dxf grayscale." A directory opens for the folder from step 1. Find and select the DXF file.
4. A new window opens containing the data from the chosen DXF file. Select the "Milster256" grayscale table and "create default." The user is thus directed back to the converter window. Change the DMD size to 400x400 pixels and click "Complete Task."
5. When the DXF file conversion is complete, click "finish."

A.3 Post-Exposure

1. Temperature plays a major role in development rate, so it is recommended to fill a beaker with 150 mL of DI water and let it sit on a hot plate set at 30°C for 5 minutes.
2. Fill a beaker with 50 mL of MicroPosit 351 developer. Add the water from step 1 to this beaker. This beaker is now a 25% developer solution.

Note: It is found that if the DI water used for development is too cold, the development rate is slower than usual, which causes under-etching and renders previous linearization steps ineffective.

3. Place the sample in a clamp and submerge it in the 25% solution for 1 minute exactly.
4. Measure the etch depths of the sample to ensure that the desired etch depth has been achieved.
5. Set a hot plate to 115°C. When it reaches the setpoint temperature, place the sample on the hot plate, photoresist side up, not contacting the hot plate. Leave the sample on the hot plate for 1 hour before removing promptly.

Note: Significant over-baking will increase the etch rate inside reactive ion etcher. Baking at too high a temperature will cause the rounding of vertical features on the photoresist.

A.4 Operating the Versaline RIE

Important: If for some reason the robotic arm cannot transfer the wafer out, it is possibly misaligned due to machine vibrations. STOP and contact the cleanroom managers. They are experienced in opening the chamber up manually remove the carrier wafer.

1. The sapphire carrier wafer should be inside the chamber when you come to use the Versaline. You can tell by the diagram on the right hand side on the GUI. If the wafer is green, the previous user likely ran an O2 clean before they left, which is the correct procedure. If the wafer is red, the an etch/clean is interrupted. Check the “versaline-status” channel for the cleanroom Slack to make sure the machine is not down and awaiting maintenance. If the wafer is dark grey, the wafer is placed in the chamber and has not been processed.
2. After logging in, click to the “Handler” tab and hit “Transfer.” You should see the robotic arm extend into the etching chamber to retrieve the wafer.
3. Vent lock, wait for the pressure in the lock to rise to 760T. You should be able to lift the lid of the lock at this point to retrieve the sapphire plate.
4. Set the sapphire plate on a flat surface. Apply thermal paste evenly to the backside of your sample and paste the sample onto the sapphire plate on the smooth side. The backside of the sapphire plate is has a grounded texture to prevent the plate from being misaligned due to vibrations inside the etching chamber. Be sure not to get thermal paste on the backside of the carrier wafer as this can cause transfer errors.

5. Place the sapphire plate with your sample pasted on back inside the lock. Click "Pump Lock." Wait for it to reach vacuum pressures, then click "Transfer." After transferring the wafer into the chamber, you will see the pressure in the lock rise to around 500 mT then automatically stabilize back down to a few mT.
6. Under the "Recipes" tab, load a "SiO2 from IMTEK" recipe. There are 2: one is "Manual Endpoint" and one is "Time-Based." Select the time-based recipe, Under the "Main Etch" step, check that the ICP power is at 1500 W and set the etch time to 10 minutes, 15 seconds. To save this customized recipe, you must save it as a personal recipe.
7. Under the "Process" tab, select the recipe you want to run, and choose a Job Id. Select "No Transfer." Start the job.
8. After the job is been completed, under the "Handler" tab, click "Transfer" and "Vent Lock."
9. The sapphire wafer will be warm to the touch so use caution when lifting the hatch and taking the wafer out. Use a razor blade to take your sample off the wafer. Set the sample to one side. Use acetone to remove thermal paste from the wafer. Be careful not to get thermal paste on the backside of the wafer because if thermal paste gets onto the electrode of the etching chamber, it can make the temperature across the sample table uneven, which will cause uneven etch rates across the surface.
10. Place the sapphire wafer back into the hatch. Pump the lock. Transfer the wafer into the chamber.
11. Under "Process," run the "O2 Clean Recipe." Make sure the "No Transfer" box is checked! The wafer needs to stay in the chamber after this clean recipe has been completed!
12. The clean recipe takes slightly over 10 minutes to complete. In the meantime, you can clean your sample with acetone to remove the thermal paste and remaining photoresist.

Fabrication Step	Key Parameters and Machine Settings
Substrate Prep	Spin-coater: • Speed: 3000 rpm • Acceleration: 1000 rotations/min/s • Duration: 30 seconds Pre-exposure bake at 120°C for 90 seconds.
Exposure	"1 in. round" substrate template 256 level grayscale table 400x400 pixel DMD
Post-Exposure	25% developer solution: • 50 ml MicroPosit 351 developer • 150 ml DI water heated to approximately 30°C Post-development bake at 115°C for 1 hour
Operating the Versaline RIE	"SiO2 from IMTEK Time-Based" recipe, main etch time 10.25 minutes, original ICP power of 1500W. See Table A2 below for recipe parameters.

Table A1. Summary of Key Parameters for the Optimal Fabrication Process

Selectivity (FS:PR)	FS Etch Rate (um/min)	PR Etch Rate (um/min)	ICP (W)	HF Bias (W)	Gas 1 C4F8 (sccm)	Gas 2 Ar (sccm)	Gas 3 O2 (sccm)	Pressure (mTorr)	Main Etch Time (min.)
1.4	0.444	0.317	1500	300	30	50	5.0	5	10.25

Table A2. Final SiO2 etch recipe. This is the original "IMTEK SiO2 recipe" with only the main etch time changed. Note that etch rates shown here are for hard-baked photoresist.

Appendix B

Documentation of FS Etch Experiments

B.1 Supplemental Figures

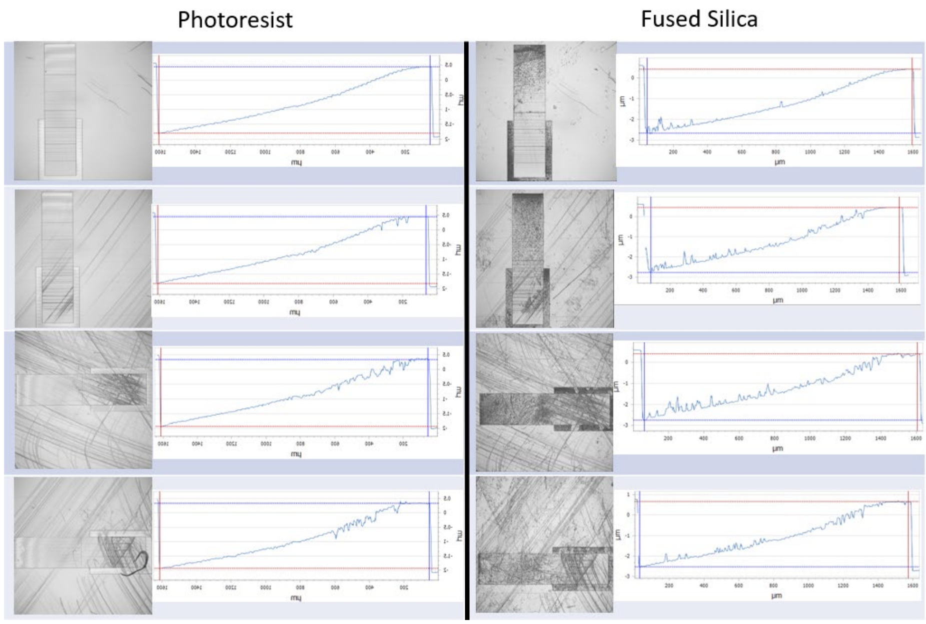
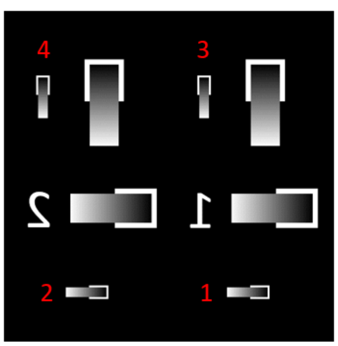


Figure B1. Fused silica etch Trial 1, on 1 mm thick substrate. There is a bit of noise in the fused silica etch, but a faithful pattern transfer between photoresist and fused silica is achieved.



Ramp #	Photoresist etch depth (um)	Fused Silica etch depth (um)	Scale FS/PR
1	2.255	3.110	1.38
2	2.281	3.216	1.41
3	2.278	3.174	1.39
4	2.265	3.177	1.40

Figure B2. Table showing etch depths measured in photoresist and in fused silica from Trial 1, showing a 1.4 scaling factor caused by the selectivity of the etch recipe.

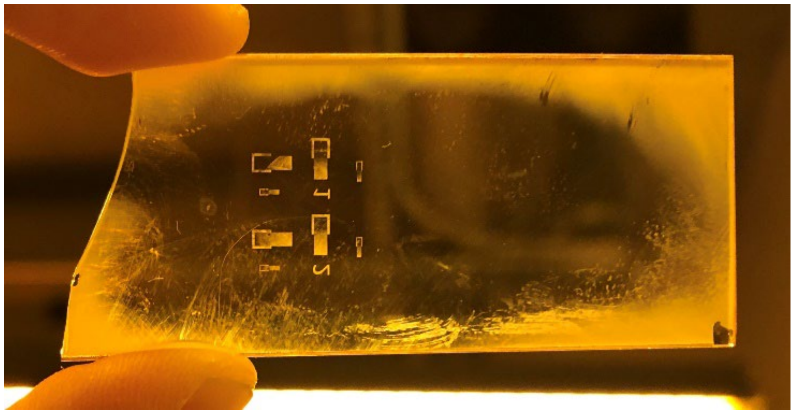


Figure B3. Photo of Trial 1 fused silica etch. It is found that the edges of the substrate clouded over, likely due to the high temperatures in the etching chamber. Applying more thermal paste, as well as turning down the ICP power help to prevent this issue.

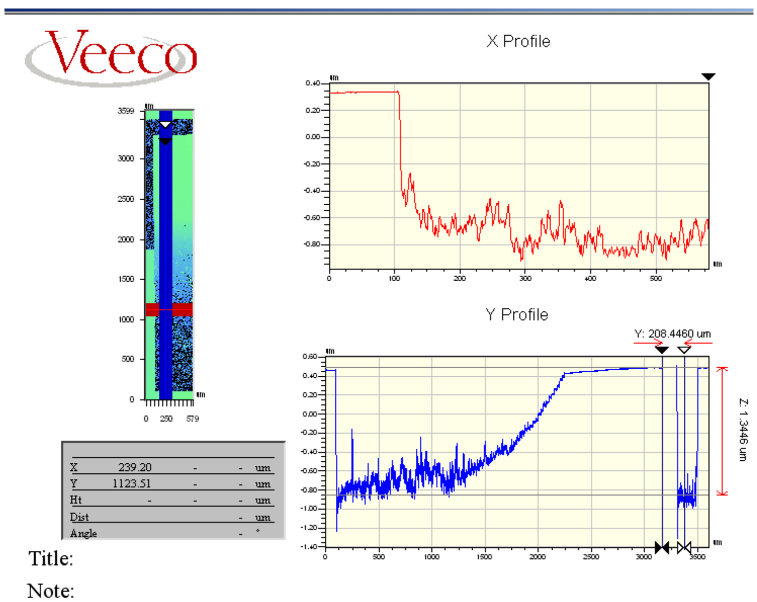


Figure B4. It is found that switching from a 1 mm substrate used for initial process development to the 2 mm substrates used to fabricate the CGH caused a significant increase in noise level.

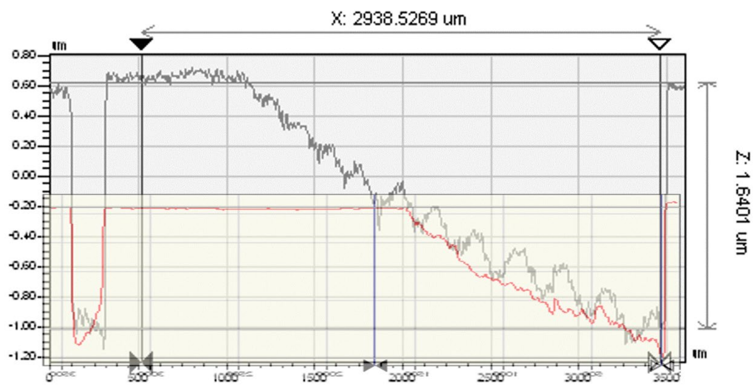


Figure B5. Relying on the interferometer on the RIE, there is always a tendency to under-etch. However, comparing the profile data from photoresist (in black) with the data from fused silica (in red) allowed for an estimation of how much the photoresist is under-etched by, and the estimated etch rate of the recipe.

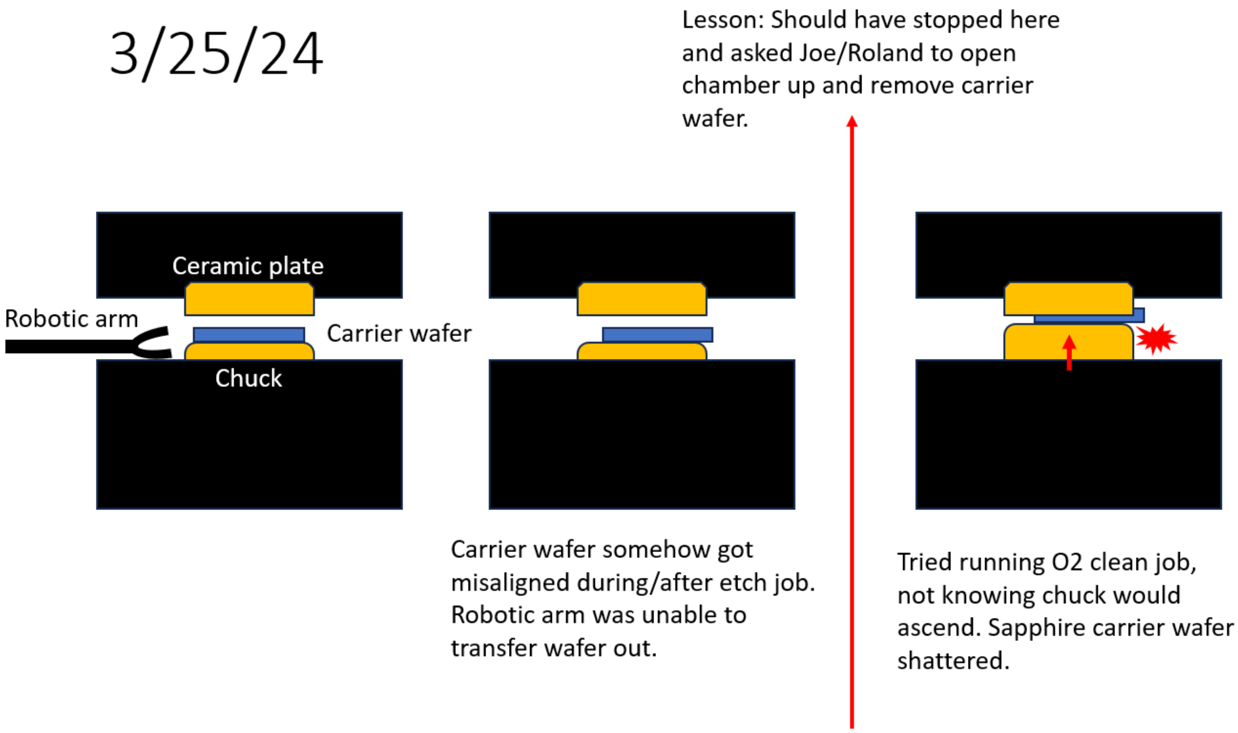
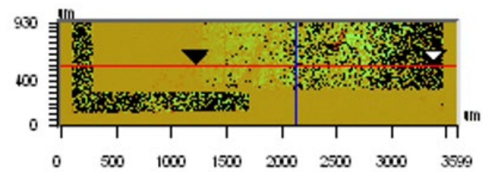


Figure B6. An important lesson is learned on 3/25/24. When operating complicated machinery, it is important to have a basic understanding of its inner workings, and when unexpected things happen, stop and ask for help from more experienced individuals.

Figure B7. Result of a photoresist-only RIE experiment. The photoresist blemished after being exposed to high temperatures inside the Versaline. The ICP power used is 1750 W, and the etch duration is 6 minutes.

Figure B8. “Process” tab in user interface of Plasma-Therm Versaline. Photo captured during a manual-endpoint etch job.

B.2 Summary of Fabrication Process Development Experiments

<table>
<tr><th>Process Variation</th><th>Effect</th></tr>
<tr><td>Etching in intervals (cooling outside chamber):
Etched a 2 mm thick, 1 in. diameter sample in three 3:45 minute intervals at 1500 W. The carrier wafer holding the sample is removed from the RIE machine and left to cool at room temperature in between etch intervals.</td><td>The final result is that the correct etch depth is achieved because the cumulative etch time between the 3 intervals had been sufficient. However, there are very noisy surface characteristics.

</td></tr>
<tr><td>Etching in intervals (cooling inside chamber):
Both experiments involved a 2 mm thick, 1 in. diameter substrate with the ICP power at 1000W. In the first experiment, a loop step is inserted into the recipe to iteratively etch the sample for 1 minute, then cool for 1 minute, for a cumulative etch time of 13 minutes. In the second experiment, the sample is etched for 7.5 minutes, cooled for 5 minutes, then etched for 7.5 minutes for a cumulative etch time of 15 minutes. The cooling steps for both experiments are added to the recipe and therefore done within the chamber. The electrode is cooled to facilitate the transfer of heat from the carrier wafer to the electrode.</td><td>For the first experiment, the photoresist at the center of the sample is visibly underetched compared to the edge, which is why the cumulative etch time is increased for the second experiment. During the cooling phase, the sample cools unevenly, which explains the difference in etch rate across the sample. For both experiments, the center ramps are much noisier compared to the ramps at the edge of the sample.</td></tr>
<tr><td>Cooling the electrode down to -10°C:
This is explored as a method to prevent the photoresist from overheating in the etching chamber. Experiments that implemented this process variation also involved cooling intervals within the chamber. Details about how to cool the electrode are discussed in the below in subsection B.3 of this appendix.</td><td>As stated above in the result for etching in intervals and cooling inside the chamber, this is not effective and caused large etch rate differences across the sample.</td></tr>
<tr><td>Fomblin PFPE lubricant for pasting the sample to the carrier wafer:
PFPE lubricant is explored as an alternative to thermal paste for the purpose of securing a given sample to the sapphire carrier wafer. The thinking is that this method will increase heat transfer between the sample and the carrier wafer, thus</td><td>No significant effect is observed for this process variation. However, it is observed that the lubricant is not an effective way to secure the sample to the wafer. To avoid having the sample possibly slide off the carrier wafer while inside the chamber, this process variation is only attempted once and subsequently abandoned.</td></tr>
</table>

preventing the photoresist from overheating. Using a Q-tip, a few drops of lubricant are applied to the carrier wafer. The sample is then placed on top of the drops and pressed to spread the lubricant out under the sample.	
Decreasing the ICP power in recipe: This is explored as the primary strategy to prevent the photoresist from overheating throughout the project. The ICP power in the original power is 1500 W, and experiments are conducted at ICP powers ranging from 750 W to 1750 W.	Experiments with only photoresist on a 2 mm thick substrate showed that decreasing the ICP power is promising in reducing noise. A sample is etched at 1000W, and it is found that the noise level is still unacceptable. Ultimately, it is determined that hard-baking the photoresist rendered decreasing the ICP power unnecessary.
Hard-baking (post-development baking): After developing the exposed photoresist, it is baked for 1 hour at 115°C on a hot plate.	Baking the photoresist helps it withstand the high temperatures inside the reactive ion etcher [3] and eliminates the precence of noise structures in the etched sample. This process variation is included in the final fabrication process.

Table B1. Fabrication process variations explored in the project and their corresponding observed effects

B.3 Procedures for Cooling the Electrode

1. Place the carrier wafer with the sample inside the chamber as usual.
2. To the left side of the etch chamber is a tube for the nitrogen pump. In the off position, the valve head should be perpendicular to the tube. Turn the valve 90 degrees clockwise so that the valve head is now parallel to the tube. The nitrogen pump is now on.
3. In the "<Initial>" stage of a given recipe, uncheck "Use Current" next to the electrode setpoint temperature, and set it to a desired temperature, e.g. -10°C. Save a personal copy of the recipe.
4. Under the "Process" tab, select the saved recipe from step 2 and click "Set Recipe Temps." In the "Temperature (°C)" quadrant in the top left of the screen, under the "Setpoint" column, the setpoint should now have changed to be the one specified the recipe. Wait for the temperature in the "Actual" column to match that of the setpoint.
5. Run the etch job as usual.
6. Under the "Process" tab, select the "DSE fast" recipe and click "Set Recipe Temps." The temperature setpoint for the electrode should now be 15°C. Wait for the actual electrode temperature to rise to the setpoint. The sound of the nitrogen pump will become more noticeable during this step.
7. Turn off the nitrogen pump by turning the valve 90 degrees counterclockwise. Now the nitrogen pump is off.
8. Remove the carrier wafer from the chamber as usual.

Appendix C

Summary of Software Packages

This appendix presents a summary of the software packages used for this project.

	Scripts	Description
Simulated Annealing Optimizer	Appendix D.1: Implementation Script Appendix D.2: Objective Function	The simulated annealing optimizer built into MATLAB is explored as an option for the second-level optimization of CGH designs. The entire optimization package consists of 2 scripts, one for calling the simulated annealing optimizer, and one to serve as the objective function. To use the optimizer, in the script provided in Appendix D1, modify the variables for the source matrix filepath and CGH design filepath. Then run the script.
Gerchberg-Saxton Design Package	Appendix D.3: G-S Main Script	Outer loop, calls Gerchberg-Saxton algorithm repeatedly to generate different CGH designs. To use this design package, provide the correct filepaths for the target image file and source distribution matrix at the top of the script before running it. Note that the for loop can be changed to "parfor" to implement parallel computing.
	Appendix D.4: G-S Algorithm	Main Gerchberg-Saxton algorithm written by Milster. Is modified by Milster on 3/8/24 to include ROI/RONI
	Appendix D.5: Design Evaluator	Automatically calculates the diffraction efficiency and uniformity of a given CGH design. Merit function is modified by Milster on 3/8/24
	Appendix D.6: Nonlinear Simulation	Simulates what the image would look like if the CGH design are fabricated with non-linearity
	Appendix D.7: Timestamp	Generates timestamp used to label folders and CGH designs
Linearization Package	Appendix D.8: Process Linearization	Appendix D.8 is used to process linearization data, or 2D etch depth data

	Appendix D.9: Linearize	for a test ramp. The script has a user interface in the command window. Upon running the script, a window pops up for the user to select their etch depth data. Then the user should flip the data as necessary so that the ramp slopes downward from left to right. The data then needs to be trimmed to be representative of the full exposure range. Then, fit a curve to the exposure curve and process the linearization. Appendix D.8 produces a calibration curve file and a linear map file. These are used in the script provided in Appendix D.9 to linearize a given CGH design. Both scripts are written by the Milster Group.
BMP to DXF Conversion	Appendix D.10: BMP to DXF Converter	This script converts a given BMP file to a DXF file so that it can be imported into the MLA 150. To use the script, provide the filepath to the input file, and change the filepath and filename of the output file as desired. Change the pixel size in microns if needed. To do this, change the "scale" variable, as well as two parameters in the "Polylinemid" vector. An example is provided in the code for which two parameters to change. This script is written by the Milster Group.

Table C1. Index of software packages provided in Appendix D.

Appendix D

Software Packages

D.1 Simulated Annealing Implementation Script

```
src_filepath     = "Src_Matrix_1000x1000_031424"; % src mask filepath
S               = load(src_filepath);  %Proportional to field of illumination.
Umask           = S.Src_Matrix;
sz              = 1000;
Q_max           = 255;

CGH_filename = "CGH_14-Mar-2024_201110_392.bmp";
x0 = imread(CGH_filename);   % Starting point
x0 = double(x0);
x0_save = x0;

flag = 4; % chosing output of objective function
type objective_func;
ObjectiveFunction = @ (X)objective_func(X, Umask, flag);

% Bounded Simulated Annealing Optimization
lb = ones(sz)*0;
ub = ones(sz)*Q_max;
options = optimoptions(@simulannealbnd,'MaxIterations',1000, ...
    'AnnealingFcn','annealingfast', 'InitialTemperature', 2000, ...
    'TemperatureFcn', 'temperatureexp', 'PlotInterval', 2, 'PlotFcn','saplotf', ...
    'ReannealInterval',100);
[X,fval,exitFlag,output] = simulannealbnd(ObjectiveFunction,x0,lb,ub,options);
X = round(X); % exposure values need to be integers
fprintf('Initial function value was: %g\n', objective_func(x0, Umask,flag));
fprintf('The best function value found was : %g\n', objective_func(X,Umask,flag));

% x0 = X; % set new starting point

% Note:
% We want to minimize "Norm. std dev" and "PV uniformity", and maximize "sum"
% By setting flag = 4, we are minimizing PV uniformity.
fprintf("Pre-Optimization\n");
fprintf('Norm. Std Dev: %g\n', objective_func(x0_save,Umask,3));
fprintf('PV Uniformity: %g\n', objective_func(x0_save,Umask,4));
fprintf('Sum: %g\n', objective_func(x0_save,Umask,5));

fprintf("Optimized\n");
fprintf('Norm. Std Dev: %g\n', objective_func(X,Umask,3));
fprintf('PV Uniformity: %g\n', objective_func(X,Umask,4));
fprintf('Sum: %g\n', objective_func(X,Umask,5));
```

D.2 Simulated Annealing Objective Function

```
function y = objective_func(X, Umask, flag)

design_dim = 1000;
Q_max = 255;

% FFT to get to image plane
CGH_mat = exp(2*pi*1i*double(X)/Q_max);
CGH_mat = CGH_mat.*Umask;
Image_test = abs(newfft2(CGH_mat)).^2;

% Find row and col locations of orders
row_save = [339-1 339+1];
col_save = [424:8:576];

output_vec = zeros(1,5);
DE_orders = calc_DE(Image_test, row_save, col_save, design_dim);
PV_uni = (max(DE_orders)-min(DE_orders))/mean(DE_orders);
output_vec(1:5) = [mean(DE_orders), std(DE_orders), ...
    std(DE_orders)/mean(DE_orders), PV_uni, sum(DE_orders)];

y = output_vec(flag); %set to standard deviation

    function DE_orders = calc_DE(Image_test_select, row_save, col_save, design_dim)
    % Isolate individual signal orders with gaussian mask
    % Calc DE of each order
    DE_orders = zeros(1,20);
    y_val_avg = (row_save(1,1)+row_save(1,2))/2;
    for num = 1:20
        x_val = (design_dim/2+1)-col_save(1,num)-1;
        y_val = (design_dim/2)-y_val_avg;
        Xmask = suprgaus(design_dim,3,y_val, x_val,10);
        Xfilter = Xmask.*Image_test_select;

        % Calculate DE of order
        Pow_orderN = sum(Xfilter(:));
        Pow_total = sum(Image_test_select(:));
        DE_orders(1,num) = Pow_orderN/Pow_total;
    end

end

end
```

D.3 Gerchberg-Saxton Main Script

```
%%%%% SETTINGS %%%%%
clear all;
close all;
clc;
tic

rng("shuffle");

% N = 4*2*4*64; % number of GS iterations
N = 1;

% Find best design with:
% 7 mean (want max)
% 8 std dev (want min)
% 9 std dev/mean (want min)
% 10 uniformity (want min)
% 11 sum (want max)
% 17 merit function (want to minimize)
choice = 17;
min_max = 1; % 1 to find min, 2 to find max
threshold = 1; % merit function threshold
% will not save design if it's above merit function threshold

%%%%%%%%%%%%%%%%%%%%
% Set Quantization levels
Q_levels = 256; % not Q_max!
% Size of CGH
N_FFT = 1000;
%%%%%%%%%%%%%%%%%%%%
% Load in source mask and object bmp

X_img            = imread("Target_1000by1000_031424.bmp",'bmp');
src_filepath     = "Src_Matrix_1000x1000_031424.mat";
S                = load(src_filepath);  %Proportional to field of illumination.
Umask            = S.Src_Matrix;
Umask            = sqrt(Umask); % you need this!

%%%%%%%%%%%%%%%%%%%%
% Read in non-linear curve for non-linearity simulation
% Ramp = csvread("Horiz_large_ramp from 80 mJ 100523A.CSV");
% Ramp = Ramp(318:1347,2)';
% Ramp = Ramp-min(Ramp);
% Ramp = Ramp/max(Ramp);
%%%%%%%%%%%%%%%%%%%%

start_time = timestamp();
% desktop_dir = 'C:\Users\wuyiy\OneDrive\□□\';
% folderName = '21-Feb-2024_100652_10000iterations/';%append(start_time, '_',num2str↙
```

```
(N),'iterations/');
folderName = append(start_time, '_',num2str(N),'iterations/');

mkdir(folderName);
output_matrix = zeros(1,17);
% init_out = zeros(1,N_FFT);
% col 1 for design #
% col 2-6 are for +5% etch depth
% col 7-11 for ideal etch depth
% col 12-16 for -5% etch depth
% col 17 merit func value

for a = 1:N % N is number of runs of G-S algorithm
    disp(['Working on trial ' num2str(a)])
    CGH_filename = append(folderName, 'CGH_',start_time, '_', num2str(a), '.bmp');

    % Run GS and find design
    [design, rand_phase] = Run_GS_YW_TM_030824(Umask ,X_img, Q_levels, N_FFT);
    % [design, rand_phase] = Run_GS_YW(Umask ,X_img, Q_levels, N_FFT);

    % Calculate DE of design
    output_vec = DE_Calc_Auto_TM_030824(a, design, Umask, 0, 1, Q_levels, N_FFT);

    % Save design if merit func below threshold
    if output_vec(17) < threshold
        imwrite(design, CGH_filename);
        % init_out = [init_out; rand_phase];
        disp(['MF < threshold for iteration ' num2str(a) ', MF = ' num2str(output_vec↙
(17))])
    end

    output_matrix = [output_matrix; output_vec];
end

% Delete first row of zeros
output_matrix = [output_matrix(2:N+1,:)];
save(append(folderName,"output_matrix.mat"),"output_matrix");
% save(append(folderName,"init_matrix.mat"),"init_out");

% Find best design
if min_max == 1
    found_val = min(output_matrix(:,choice));
elseif min_max == 2
    found_val = max(output_matrix(:,choice));
end
for a = 1:N
    if (output_matrix(a,choice)==found_val)
        design_num = output_matrix(a,1);
    end
end
```

```
fprintf("\nBest Design is Trial #%d\n", design_num);

% Going to pop up figures for the best design
chosen_design = append(folderName, 'CGH_',start_time, '_', num2str(design_num), '.↙
bmp');
design = imread(chosen_design,'bmp');
DE_Calc_Auto_TM_030824(0, design, Umask, 1, 1, Q_levels, N_FFT);
% nonlinear_sim(design, Umask, Ramp, 1, Q_levels, N_FFT);
start_time
output_matrix(design_num,:)
x = output_matrix(:,11);
y = output_matrix(:,10);
figure; plot(x,y,'.'); title("PV Uniformity vs. DE");

toc
```

D.4 Gerchberg-Saxton Algorithm

```
function [design, rand_phase] = Run_GS_YW_TM_030824(Umask, X_img, Q_levels, N_FFT)
%030824 - Includes ROI/RONI
close all;
N_Loops         = 1000;
Display_flag    = 0;
% horiz_range     = 2.56*pi/180;  %range in radians for horizaontal dimension of image↙
(input is deg, must change to rad)
% LAMBDA          = 6.5e-7; % switched to supercont laser
% n_resist        = 1.62;
% n_incident      = 1;
% GS_max_OPD      = LAMBDA/(n_resist-n_incident);   %Default is one lambda of OPD in↙
the center
thresh          = 0.01;
Mask_flag       = 1; %we are using a source mask!
% address_unit    = 15e-6/2; %scaled down from design 1 AU of 29 um
fid             = 1;


% disp(['Size of DOE element = ' num2str(N_FFT*address_unit) ' meters in horizontal↙
dimension.'])
%determine sampling vectors
[nr nc]         = size(X_img);
% max_FFT_range   = asin(LAMBDA/address_unit/2)*2;   %111514 Modified w asin and factor↙
of 2 TDM %020524 added /2
% xvec_img_old    = linspace(-horiz_range/2,horiz_range/2,nc);
% yvec_img_old    = linspace(-horiz_range/2*nr/nc,horiz_range/2*nr/nc,nr);
% xvec_img_new    = linspace(-max_FFT_range/2,max_FFT_range/2,N_FFT);
% yvec_img_new    = xvec_img_new;
% xsample_new     = xvec_img_new(2)-xvec_img_new(1);


%Define region of interest (ROI) index
ROI         = 0.75;
ROI_mask    = [ones(round(ROI*nr),nc); zeros(nr-round(ROI*nr),nc)];%suprgaus(nr,nr/2*0.↙
9,0,0,75);
ROI_indx    = find(ROI_mask>0.1);
RONI_indx   = find(ROI_mask<=0.1);
N_ROI       = N_Loops-7;  %number of RONI cycles
```

```
I_img = double(X_img);
indx              = find(isnan(I_img));
I_img(indx)       = zeros(1,length(indx));
P_img    = sum(I_img(:));

%calculate window for estimate space during the calculation
window            = (0.5+0.5*cos(pi*linspace(-1,1,N_FFT)))'*(0.5+0.5*cos(pi*linspace↙

 (-1,1,N_FFT)));
 window_round     = suprgaus(N_FFT,N_FFT/2-1);

 %Initialize with random phase in  Fourier plane
 % disp('Initializing phase');
 rand_phase = rand(N_FFT);
 U_work           = (I_img).^(0.5) .*exp(1i*2*pi*rand_phase);

 % Process source distribution mask
 [nrX ncX]        = size(Umask);

 %Check Quantize Level
 if ~(Q_levels>1)
     disp('***** ERROR:  Inappropriate quantization level')
     error('Inappropriate quantization level')
     return
 end

 %performing GS algorithm
 skip_Q       = 1;
 for ii = 1:N_Loops
     %disp(['Working on GS loop ' num2str(ii)])
     if ii == 1
         SNR_vec = [];
     end
     U_DOE        = newifft2(U_work);
     if Q_levels == 2
         Q_angle      = pi*round(abs(angle(U_DOE))/pi);
     else
         Q_angle      = 2*pi*round((Q_levels-1)*(angle(U_DOE)+pi)/2/pi)/(Q_levels-1);
     end
     %NEW SECTION  FOR PSF%%%%%%%%%%%%% 052511
     %filter for convolution
 %              max_Q        = max(max(Q_angle));
 %              min_Q        = min(min(Q_angle));
 %              v            = suprgaus(5,1,0,0,1);
 %              Q_angle      = conv2(rot90(Q_angle,3),v,'same');
 %              max_Q2       = max(max(Q_angle));
 %              min_Q2       = min(min(Q_angle));
 %              Q_angle      = (Q_angle-min_Q2)/(max_Q2-min_Q2)*(max_Q-min_Q);
     %%%%%%%%%%%%%%%%%%%%%%%%%%
     if Mask_flag  %added 081314 for illumination
         U_DOE   = ones(size(U_DOE)).*exp(1i*Q_angle).*Umask;%.*window; %REMOVED WINDOW↙
 051511 TDM
     else
         U_DOE   = ones(size(U_DOE)).*exp(1i*Q_angle);%.*window; %REMOVED WINDOW 051511↙
 TDM
     end
     U_work       = newfft2(U_DOE);
     %U_work       = newfft2(U_DOE.*window_round); %option to add round window 101613 TDM
```

```
    I_work      = abs(U_work).^2/max(max(abs(U_work).^2));
    %figure(200);imagesc(I_work(350:650,350:650));pause(1)
    if Display_flag == 1 && ii >1 %% SNR Graph!
        index_inside_ideal  = find(I_img>=thresh);   %This vector indicates the
 locations in the ideal matrix that are at or above threshold.
        index_outside_ideal = find(I_img<thresh);  %This vector indicates the locations
 in the ideal matrix that are below threshold.
        SNR                    = 10*log10( sum(I_work(index_inside_ideal) ) / sum( I_work
 (index_outside_ideal) ) );  %SNR in dB
        SNR_vec                = [SNR_vec SNR];
        figure(fid);plot(SNR_vec,'o-');xlabel('Iteration Number');ylabel('SNR in dB');
 grid;pause(0.1)
        set((fid),'Name','DOE GENERATOR: GS Algorithm')
 %                figure;imagesc(abs(I_work));colormap(gray);title(['Image Distribution
 - itteration ' num2str(ii)]);axis image
 %                figure;imagesc(abs(Q_angle));colormap(gray);title(['Phase
 Distribution - itteration ' num2str(ii)]);axis image
 %                pause(1)
    end
    if Display_flag == 1 && mod(ii,10) == 0
        figure((fid+1));imagesc(abs(I_work));colormap(gray);title('Image
 Distribution');axis image
        % figure((fid+2));imagesc(angle(I_DOE));colormap(gray);title('DOE Phase
 Distribution'); colorbar
    end
     % U_work      = (I_img).^(0.5) .*exp(1i*angle(U_work));
     P_work     = sum(abs(U_work(:)).^2);
     if ii < N_ROI
        U_work(ROI_indx)    = (I_img(ROI_indx)).^(0.5) .*exp(1i*angle(U_work
 (ROI_indx)));
        U_work(RONI_indx)   = sqrt(P_img/P_work)*abs(U_work(RONI_indx)).*exp(1i*angle
 (U_work(RONI_indx)));
     else
        U_work(ROI_indx)    = (I_img(ROI_indx)).^(0.5).*exp(1i*angle(U_work(ROI_indx)));
        U_work(RONI_indx)   = sqrt(P_img/P_work)*abs(U_work(RONI_indx)).*exp(-(ii-N_ROI)
 /10).*exp(1i*angle(U_work(RONI_indx)));
 end

 % Generate and Print Exposure Map
 my_mat = (angle(U_DOE)+pi)/(2*pi); % map to range [0,1]
 my_mat = uint8(round((Q_levels-1)*my_mat/max(my_mat(:)))); % scale t0 [0,Q_levels-1]

% Function returns design matrix as output
design = my_mat;

end
```

D.5 Design Evaluator

```
function output_vec = DE_Calc_Auto_TM_030824(N, X, Umask, display_flag, mask_flag,↙
Q_levels, design_dim)
% Put 0 to check mask centering
center_good = 1;

% Find row and col locations of orders
% [row_save, col_save] = find_orders(Image_test, design_dim);
r = 339;
row_save = [r-1 r+1];
col_save = [424:8:576];

% FFT to get to image plane
Q_max = Q_levels -1;
CGH_mat = exp(2*pi*1i*double(X)/Q_max);
if mask_flag == 1
    CGH_mat = CGH_mat.*Umask;
end
Image_test = abs(newfft2(CGH_mat)).^2;


% Show figure of exposure map and image plane
if display_flag== 1
   figure; imagesc(X); colormap(gray); colorbar; title("DOE Exposure Map");
   figure;imagesc(Image_test);axis image;colormap(gray);
   title("Image Plane");
end

% max etch depth
LAMBDA = 1.064e-6;
n_resist = 1.45; % index of FS at 1064 nm
n_incident = 1;
GS_max_OPD = LAMBDA/(n_resist-n_incident);   %Default is one lambda of OPD in the↙
center

% delta_h to delta_phi conversion (have 5% etch depth tolerance)
delta_h = 0.05*GS_max_OPD;
delta_phi = 0.5/LAMBDA*delta_h;

% Depth var +5%
CGH_matP10 = exp((1+delta_phi)*2*pi*1i*double(X)/Q_max);
if mask_flag == 1
    CGH_matP10 = CGH_matP10.*Umask;
end
Image_testP10 = abs(newfft2(CGH_matP10)).^2;

% Depth var -5%
CGH_matM10 = exp((1-delta_phi)*2*pi*1i*double(X)/Q_max);
if mask_flag == 1
    CGH_matM10 = CGH_matM10.*Umask;
end
```

```
Image_testM10 = abs(newfft2(CGH_matM10)).^2;

output_vec = zeros(1,17);
output_vec(1,1) = N;
for j = 1:3
    if j == 1
        Image_test_select = Image_testP10;
        string = '+5% IN THICKNESS';
    elseif j == 2
        Image_test_select = Image_test;
        string = 'IDEAL THICKNESS';
    else
        Image_test_select = Image_testM10;
        string = '-5% IN THICKNESS';
    end
    % fprintf("\n");
    % disp(string);
    DE_orders = calc_DE(Image_test_select, row_save, col_save, center_good,
design_dim);
    orders = [-10:-1 1:10];

    if display_flag == 1
        figure; bar(orders, DE_orders);
        title("Diffraction Efficiency of Signal Orders");
        xlabel("Odd Diffraction Orders (+/- 10)");
        ylabel("DE");
        subtitle(string);
        ylim([0,0.06]);
    end

    PV_uni = (max(DE_orders)-min(DE_orders))/mean(DE_orders);

    output_vec(2+(j-1)*5:6+(j-1)*5) = [mean(DE_orders), std(DE_orders), ...
        std(DE_orders)/mean(DE_orders), PV_uni, sum(DE_orders)];
end
w = [0, 0.75, 1, 0, 0.25, 0.25, 0, 0.25 , 0.25]; % merit function weights
% err = [0.05-output_vec(7), output_vec(8), output_vec(9), output_vec(10), 1-output_vec
(11)];
err = [output_vec(9), output_vec(10), 1-output_vec(11), ...
        output_vec(4), output_vec(5), 1-output_vec(6), ...
        output_vec(14), output_vec(15), 1-output_vec(16)];
output_vec(1,17) = sqrt(sum(w.*err.^2));

    function DE_orders = calc_DE(Image_test_select, row_save, col_save, center_good,
design_dim)
    % Isolate individual signal orders with gaussian mask
    % Calc DE of each order
    DE_orders = zeros(1,20);
    y_val_avg = (row_save(1,1)+row_save(1,2))/2;
    for num = 1:20
```

```
        x_val = (design_dim/2+1)-col_save(1,num)-1;
        y_val = (design_dim/2)-y_val_avg;
        Xmask = suprgaus(design_dim,3,y_val, x_val,10);
        Xfilter = Xmask.*Image_test_select;

        if center_good == 0
            % Make sure we centered on diff order
            figure; imagesc(Xmask);
            ylim([row_save(1,1)-5, row_save(1,2)+5]);
            xlim([col_save(1,num)-20,col_save(1,num)+20]);
            % Should be centered on cross-hairs
            hold on; xline(col_save(1,num)+0.5); yline(row_save(1,1)+0.5); hold off;
            title("Order "+num);
        end

        % Calculate DE of order
        Pow_orderN = sum(Xfilter(:));
        Pow_total = sum(Image_test_select(:));
        DE_orders(1,num) = Pow_orderN/Pow_total;
    end

end

    function [row_save, col_save] = find_orders(Image_test, design_dim)
    % Look for rows with non-zero values, save row numbers
    row_prev = 0;
    count = 0;
    for a = 1:design_dim
        for b = 1:design_dim
            if Image_test(a,b) > 1e8 && a ~= row_prev
                row_save(1,count+1) = a;
                row_prev = a;
                count = count + 1;
            end
        end
    end
    row_save;

    % Look for cols with non-zero values, save col numbers
    % Save left-most col #

    count = 0;
    a = row_save(1,1);
    col_prev = 0;
    for b = 1:design_dim % just scan across a single row
        if Image_test(a,b) > 1e8 && b ~= col_prev+1
            col_save(1,count+1) = b;
            col_prev = b;
            count = count + 1;
        end

    end
    col_save;
end

end
```

D.6 Nonlinear Simulation

```
% clc
% clear all
% close all
%
% CGH_filename = "C:\Users\wuyiy\Downloads\Thesis Project\650 nm Design and↙
Testing\Design 1 Iterations\Chosen Design\CGH_04-Jan-2024_031805_207.bmp";
%
function nonlinear_sim(X, Umask, Ramp, mask_flag, Q_levels, design_dim)

% Make noise matrix to be added
Q_max = Q_levels-1;
% s = 0*Q_max/10*randn(512);
% X = double(X)+s;

% Polynomial fit
val = 1:1030;
p = polyfit(val,Ramp,3);
x1 = linspace(1,1030);
y1 = polyval(p,x1);
% figure; plot(Ramp,'.'); hold on; plot(x1,y1); axis tight; hold off;
% legend('Data','Fitted Function','Location','northwest');
% title('Data and Fitted Curve');

% Make non-linear curve 0:1
num = 1:Q_max;
non_lin = polyval(p,num*16);
non_lin = non_lin-min(non_lin);
non_lin = non_lin/max(non_lin);
lin = num/Q_max;

% Read in ramp design for testing
% R = imread("C:\Users\wuyiy\Downloads\Thesis Project\Test Ramp↙
Designs\test_ramps_small_092923.bmp");
% R = round(double(R)./4);
% R2 = R;

% Apply non-linearity
for a = 1:Q_max
    indx = find(X==a);
    X(indx) = (non_lin(a)-lin(a)+1)*ones(1,length(indx))*a;
    %indx = find(R2==a);
    %R2(indx) = (non_lin(a)-lin(a)+1)*ones(1,length(indx))*a;
end


% FFT to get to image plane
CGH_mat = exp(2*pi*1i*double(X)/Q_max);
if mask_flag == 1
    CGH_mat = CGH_mat.*Umask;
end
```

```
Image_test = abs(newfft2(CGH_mat)).^2;

% Image processing to make higher orders visible
count = 1;
for a = 1:design_dim
    for b = 1:design_dim
        if Image_test(a,b) > 1e8
            Signal_orders(count, 1) = a;
            Signal_orders(count, 2) = b;
            count = count+1;
        end
        while Image_test(a,b) > 1e4
            Image_test(a,b) = Image_test(a,b)/10;
        end
    end
end

% Show figure of image plane
figure;imagesc(Image_test); colormap(gray); axis image; title("Non-Linear Image↙
Plane"); % subtitle("Non-linearity Added");
hold on; plot(Signal_orders(:,2), Signal_orders(:,1), 'w.'); hold off;

% Figure showing how much non-linearity was added
% figure; plot(R(150,:),'.'); hold on; plot(R2(150,:),'.'); hold off; axis tight;
% title("Checking Code w/ Ramp Design BMP")
% legend("Original Design","Non-Linearity Added",'Location','southeast');

end
```

D.7 Timestamp

```
function time_now = timestamp()
time_now = datestr(datetime('now'));
time_now(time_now == '.') = [];
time_now(time_now == ':') = [];
time_now(time_now == ' ') = '_';
end
```

D.8 Process Linearization

```
function    [height_data, interface_flag_out] = process_linearization_040822(varargin)

parameters          = varargin_to_struct( varargin{:} );

height_data         = eval('parameters.height_data',            '[]');
LAMBDA              = eval('parameters.lambda',                 '405e-9');
n_resist            = eval('parameters.n_resist',               '1.5802');
n_incident          = eval('parameters.n_incident',             '1.0');
differential_etch   = eval('parameters.differential_etch',      '1');
resist_type         = eval('parameters.resist_type',            '1');
harmonic_value      = eval('parameters.harmonic_value',         '1.0');
zero_bmp_flag       = eval('parameters.zero_bmp_flag',          '0');
t_max               = eval('parameters.t_max',                  '0');
start_dir_Veeco     = eval('parameters.start_dir_veeco',        '''D:↙
\CONTRACTS\DoD\Single-Shot Phase Retrieval 2022''');
start_dir_MLT       = eval('parameters.start_dir_mlt',          '''D:↙
\CONTRACTS\DoD\Single-Shot Phase Retrieval 2022''');
project_directory   = eval('parameters.project_directory',      '''D:↙
\CONTRACTS\DoD\Single-Shot Phase Retrieval 2022''');
resist_dispersion_file = eval('parameters.resist_dispersion_file', '''D:↙
\CONTRACTS\DoD\Single-Shot Phase Retrieval 2022''');
length_end_slope_calc = eval('parameters.length_end_slope_calc','40');
length_lead_slope_calc = eval('parameters.length_lead_slope_calc','40');
fig_id              = eval('parameters.fig_id',                 '20');
save_pname          = fullfile(start_dir_MLT,'linear_map.mat');
save_hd_pname       = fullfile(start_dir_MLT,'CALIBRATION_CURVE.mat');

if isempty(height_data)
    [filename, pathname] = uigetfile('*.csv', 'Pick a csv file containing the test ramp↙
data',start_dir_Veeco);
    if isequal(filename,0)
        disp('User selected Cancel')
        return
    else
        pname           = fullfile(pathname, filename);
        disp(['User selected ', pname])
    end

    %process csv file
    fid1                = fopen(pname,'r');
    fid2                = fopen([pname(1:(end-4)) '.txt'],'w+');
    EOF_indicator       = 0;
    line                = fgetl(fid1);
    test_leading_slope_flag     = 1;
    test_line          = 0;
    while test_line ~=-1
        test_line       = fgetl(fid1);
%         index           = strfind(test_line,'---');  %this section was
%         for removing bad data from NT9800.  Removed 040822
%         for ii = 1:length(index)
```

```
%             test_line(index(ii):(index(ii)+2))='NaN';
%         end
        if isstr(test_line)
            index       = strfind(test_line,',Slice 1');
            test_line   = str2num(test_line(1:index));
        end
        if test_line ~=-1
            if test_leading_slope_flag
                if  line >= 0.9*test_line(2)
                    fwrite(fid2, [num2str(test_line(2)) ','], 'char');
                    test_leading_slope_flag     = 0;
                else
                    line         = test_line(2);
                end
            else
                fwrite(fid2, [num2str(test_line(2)) ','] , 'char');
            end
        end
    end
    fclose(fid1);
    fclose(fid2);

    height_data         = load([pname(1:(end-4)) '.txt']);
    height_data         = (height_data- min(height_data))*1e-6;
    index               = find(isnan(height_data));
    if ~isempty(index)
        disp(['*****WARNING: NaNs located in height data.  Values are set to zero at↙
locations ' num2str(index)])
    end
    height_data(index) = 0;

end
linearize_thickness= height_data;

interface_flag  = 1;
while interface_flag
    xvec         = linspace(0,100,length(height_data));
    figure(fig_id);plot(xvec,height_data);grid
    disp('*********************************')
    disp('      1. Adjust end slope.')
    disp('      2. Fit curve to data.')
    disp('      3. Flip curve LR.')
    disp('      4. Flip curve UD.')
    disp('      5. Trim along x axis.')
    disp('      6. Adjust leading slope.')
    disp('      7. Process linearization.')
    disp('      99. Up one level.')
    disp('      100. Quit.')
    go_flag         = input('What would you like to do (1, 2, 3, 4, 5, 6, 7, 99 or 100)?↙
');
```

```
    if ~isempty(go_flag) && any(go_flag==[1 2 3 4 5 6 7 99 100])

        if go_flag  == 1

            %take out end slope
            end_slope        = sum(diff(height_data((end-length_end_slope_calc):end)))/
(length_end_slope_calc-1);
            height_data      = height_data - end_slope*(1:length(height_data));
            height_data      = height_data- min(height_data);
            xvec             = linspace(0,100,length(height_data));
            figure(fig_id);plot(xvec,height_data);grid

        elseif go_flag == 2

            height_data          = fit_hight_data_curve(height_data,fig_id);

        elseif go_flag == 3

            height_data          = fliplr(height_data);
            figure(fig_id);plot(xvec,height_data);grid
            title('Processed Curve')
            xlabel('Laser Power in %')
            ylabel('Depth in um')

       elseif go_flag == 4

            height_data          = max(height_data(:)) - height_data;
            figure(fig_id);plot(xvec,height_data);grid
            title('Processed Curve')
            xlabel('Laser Power in %')
            ylabel('Depth in um')

       elseif go_flag == 5

            disp('***** Use magnifier to dertermine start and end points before
continuing. *****')
            startingx            = input('Enter starting x value in micrometers : ');
            endingx              = input('Enter ending x value in micrometers : ');
            height_data          = height_data(xvec>=startingx & xvec<=endingx);
            xvec                 = linspace(0,100,length(height_data));
            figure(fig_id);plot(xvec,height_data);grid
            title('Processed Curve')
            xlabel('Laser Power in %')
            ylabel('Depth in um')

        elseif go_flag  == 6

            %take out leading slope
            lead_slope       = sum(diff(height_data(1:length_lead_slope_calc)))/
(length_lead_slope_calc-1);
```

```
            height_data     = height_data - lead_slope*(1:length(height_data));
            height_data     = height_data- min(height_data);
            xvec            = linspace(0,100,length(height_data));
            figure(fig_id);plot(xvec,height_data);grid
            title('Processed Curve')
            xlabel('Laser Power in %')
            ylabel('Depth in um')

        elseif go_flag == 7

            save CALIBRATION_CURVE height_data
            save(save_hd_pname,'height_data')
            [harmonic_value,t_offset,interface_flag] = choose_offset(harmonic_value,↙
LAMBDA,n_resist,n_incident,differential_etch,t_max,...
                fig_id,save_pname,[],height_data,resist_type,resist_dispersion_file,↙
zero_bmp_flag);
            if interface_flag == 0
                interface_flag_out  = 0;
            end

        elseif go_flag == 99

           interface_flag       = 0;
           interface_flag_out   = 1;

        elseif go_flag == 100

           interface_flag       = 0;
           interface_flag_out   = 0;

        end

    end
end

end
%***************************************************************************
%***************************************************************************
%                           Subroutines
%***************************************************************************
%***************************************************************************

%***************************************************************************
%                   Choose offset
%***************************************************************************
function [harmonic_value,t_offset,interface_flag_out] = choose_offset(harmonic_value,↙
LAMBDA,n_resist,n_incident,differential_etch,t_max,...
    fig_id,save_pname,xvec,height_data,resist_type,resist_dispersion_file,↙
zero_bmp_flag)
```

```
if t_max == 0
    t_max           = LAMBDA*harmonic_value/(n_resist-n_incident)*differential_etch;
end
show_flag                       = 1;
fid                             = fig_id;
disp('LINEARIZE EXPOSURE ***********************************************************')
if ~exist('t_offset')
    t_offset                    = 0;
end
if ~exist('xvec') || isempty(xvec)
    xvec                = linspace(0,100,length(height_data));
end

interface_flag_out  = 1;
interface_flag = 1;
while interface_flag
    indx                        = find(height_data>=t_offset & height_data<=↙
(t_max+t_offset));
    figure(fig_id+2);plot(xvec,height_data,'k',xvec(indx),height_data(indx),'go')
    xlabel('Exposure Percentage')
    ylabel('Height (m)')
    legend('Calibration curve','Exposure Range')
    grid;pause(0.1)
    if (t_max+t_offset)>max(height_data)
        disp('*****WARNING: Requested height too large for calibration.  Reduce offset↙
and/or harmonic value.')
    end
    diff_height             = diff(height_data(indx));
    if any(diff_height)==0
        disp('*****WARNING: Non-unique height data in requested range.  Increase↙
offset.')
    end

    disp↙
('*********************************************************************************')
    disp(['      1. Choose new harmonic value (presently ' num2str(harmonic_value)↙
').'])
    disp(['      2. Chose new offset (presently ' num2str(t_offset) ').'])
    disp(['      3. Chose new wavelength (presently ' num2str(LAMBDA) ').'])
    disp(['      4. Chose new resist refractive index (presently ' num2str(n_resist)↙
').'])
    disp(['      5. Chose new incident refractive index (presently ' num2str↙
(n_incident) ').'])
    disp('      6. Save calibration curve.')
    disp(['      7. Force modulator zero on MLT outside of print area? (Y = 1,↙
presently ' num2str(zero_bmp_flag) ').'])
    disp('     99. Up one level.')
    disp('     100. Quit.')
    disp(['    ***** Maximum resist sag for deisgn = ' num2str(t_max) ' m *****'])
    go_flag         = input('What would you like to do (1-7, 99, 100)? ');
```

```
    if ~isempty(go_flag) & any(go_flag==[1 2 3 4 5 6 7 99 100])

        if go_flag  == 1

            %choose new harmonic value
            new_harmonic_value  = input('Enter new harmonic value : ');
            if ~isempty(new_harmonic_value)
                harmonic_value  = new_harmonic_value;
            end
            t_max                = LAMBDA*harmonic_value/(n_resist-n_incident)↙
*differential_etch;
            disp(['Maximum resist sag for deisgn = ' num2str(t_max) ' m'])

        elseif go_flag == 2

            new_t_offset         = input('Enter new offset value (units = m) : ');
            if ~isempty(new_t_offset)
                t_offset         = new_t_offset;
            end

        elseif go_flag == 3

            %choose new wavelength value
            new_wavelength_value  = input('Enter new wavelength value (m) : ');
            if ~isempty(new_wavelength_value)
                LAMBDA   = new_wavelength_value;
                resist_index_mat = load(resist_dispersion_file);
                n_resist          = interp1(resist_index_mat(:,1),resist_index_mat(:,2),↙
LAMBDA);
            end
            t_max                = LAMBDA*harmonic_value/(n_resist-n_incident)↙
*differential_etch;
            disp(['Maximum resist sag for deisgn = ' num2str(t_max) ' m'])

        elseif go_flag == 4

            %choose new resist refractive index value
            new_resist_refractive_index_value   = input('Enter new resist refractive↙
index value : ');
            if ~isempty(new_resist_refractive_index_value)
                n_resist   = new_resist_refractive_index_value;
            end
            t_max                = LAMBDA*harmonic_value/(n_resist-n_incident)↙
*differential_etch;
            disp(['Maximum resist sag for deisgn = ' num2str(t_max) ' m'])

        elseif go_flag == 5

            %choose newincident refractive index value
            new_incident_refractive_index_value   = input('Enter new incident refractive↙
```

```
index value : ');
            if ~isempty(new_incident_refractive_index_value)
                n_incident  = new_incident_refractive_index_value;
            end
            t_max                = LAMBDA*harmonic_value/(n_resist-n_incident)↙
*differential_etch;
            disp(['Maximum resist sag for deisgn = ' num2str(t_max) ' m'])

        elseif go_flag == 6

            process_data_and_save(height_data,t_max,t_offset,fig_id,save_pname,↙
resist_type,zero_bmp_flag);

        elseif go_flag == 7

            %choose zero bmp flag
            zero_bmp_flag  = input('Force zero bmp level outside of print area (Y ==↙
1)? : ');
            if isempty(zero_bmp_flag)
                zero_bmp_flag   = 0;
                disp('Modulator zero will not be forced outside of print area.')
            elseif zero_bmp_flag ~= 1
                zero_bmp_flag   = 0;
                disp('Modulator zero will not be forced outside of print area.')
            else
                disp('Modulator zero will be forced outside of print area.')
            end

        elseif go_flag == 99

            interface_flag      = 0;
            interface_flag_out  = 1;

        elseif go_flag == 100

            interface_flag      = 0;
            interface_flag_out  = 0;

        end

    end

end

end
%************************************************************************
%                            exposure_fit_error
%************************************************************************
function error_result = exposure_fit_error(x,xvec,height_data)
```

```
% exposure_curve      = @(x) x(1)*exp(x(2)+x(3)*0  - (x(4)*xvec.*xvec).^x(5)) + x(6)↙
*0*xvec.*exp(-(x(7)*xvec.*xvec).^x(8) + 0*x(9));

difference_data     = exposure_curve(x,xvec) - height_data';
if min(difference_data)~=difference_data(end)
    error_result    = 1000;
    %disp('E1')
end

if min(difference_data)~= min(height_data)
    error_result    = 1000;
    %disp('E2')
end

error_result        = sum( (difference_data).^2 );
if ~isreal(error_result)
    error_result    = 1000;
end

if any(diff(exposure_curve(x,xvec))>0)
    indx            = find(diff(exposure_curve(x,xvec))>0);
    error_result    = error_result + sum(difference_data(indx).^2);
end
end
%***************************************************************************
%                           exposure_curve
%***************************************************************************
function y = exposure_curve(x,xvec)

y           = x(1)*exp(x(2) - (x(3)*xvec.*xvec).^x(4)) + x(5)*exp(x(12)-(x(6)*xvec.↙
*xvec).^x(7)) + x(8)*exp(x(13)-(x(9)*xvec.*xvec).^x(10)) + x(11);
index       = find(y<0);
y(index)    = 0;
end
%***************************************************************************
%                           fit_height_data_curve
%***************************************************************************
function   height_data          = fit_hight_data_curve(height_data,fig_id);

%now fit curve
disp('This program calculates a fit to the height_data vector in a mat file according↙
to:')
disp(' fit function =  x(1)*exp(x(2) - (x(3)*xvec.*xvec).^x(4)) + x(5)*exp(x(12)-(x(6)↙
*xvec.*xvec).^x(7)) ...')
disp('                 + x(8)*exp(x(13)-(x(9)*xvec.*xvec).^x(10))')
disp('The program uses the fminsearch Matlab function, so it has a tendency to get ')
disp('  stuck in local minima.  Twenty five random perturbations of the starting vector↙
')
disp('  are used to attempt a more global optimum.  The user can input a starting ')
disp('  vector, if desired.  The height_data vector corresponds to resist thickness,↙
```

```
and it  ')
disp('  starts at zero exposure and ends at 100% exposure.  Thickness has units of↙
meters.')
disp('  There is no restriction on vector length, but it should have at least 50↙
points.')
disp('  If the fit is not very well, you can rerun the routine automatically.')
pause(1)
disp(' ')

if ~(exist('height_data')==1)
    disp('*****ERROR: mat file does not contain a vector named height_data.')
    return
elseif any(~isreal(height_data))
    disp('*****ERROR: Imaginary data in height_data.')
    return
elseif length(height_data(:)) < 3
    disp('*****ERROR: Please use more data points for height_data.')
    return
end

height_data          = height_data(:);

x0_default           = [1.6e-007 7e-005 7e-008   3 4.5e-007 3e-005   1.4   9e-007   1.5e-↙
005  4 -1.6e-007 0 0];

repeat_flag          = 1;

new_x0_flag          = 0;
x0                   = x0_default;

xvec                 = 1:length(height_data);
height_data_scaled   = height_data;

fval_min_base        = 1.1*exposure_fit_error(x0_default,xvec,height_data);
fval_min             = fval_min_base;
while repeat_flag == 1

    %exposure_curve       = @(x)  x(1)*exp(x(2)+x(3)*0   -  (x(4)*xvec.*xvec).^x(5)) + x(6)↙
*0*xvec.*exp(-(x(7)*xvec.*xvec).^x(8) +0*x(9));
    x                    = x0;
    disp('**************************************')
    disp('Start data fitting routine')
    disp('Please wait for processing to complete')
    %     keyboard

    for jj = 1:1
        front_factor             = 0.25;
        for ii = 1:25
            x0_trial             = x + front_factor*(0.5*rand(1,length(x0)) - 1).*x;
            indx                 = find(x0_trial <0);
```

```
            x0_trial(indx)       = zeros(1,length(indx));
            [x_trial, fval, exitflag] = fminsearch(@exposure_fit_error, x0_trial,...
                optimset('TolX',1e-7,'MaxFunEvals',500,'Display','off'),xvec,↙
height_data_scaled);
            if fval < fval_min & isreal(fval)
                fval_min         = fval;
                x_min            = x_trial;
                x                = x_trial;
                y_test           =  exposure_curve(x,xvec);
                figure(fig_id);plot([height_data';y_test]');grid
                disp(['NEW MINIMUM ERROR = ' num2str(fval)])
                front_factor     = front_factor*0.5;
            end
        end
    end

    disp(['Minimum error = ' num2str(fval_min)])
    disp(['Optimized fit vector = ' num2str(x_min)])
    repeat_flag              = input('Would you like the routine to try again (Y = 1) ?↙
');
    if repeat_flag
        %reset_flag           = input('Would you like to reset the coefficient vector (Y↙
= 1) ? ');
        if 1 %reset_flag
            %new_x0_flag         = input('Would you like to enter your own starting↙
values (Y = 1) ? ');
            new_x0_flag      = 0;
            if ~isempty(new_x0_flag) & new_x0_flag == 1
                disp('Enter the starting vector')
                x0(1)            = input('Enter x1:');
                x0(2)            = input('Enter x2:');
                x0(3)            = input('Enter x3:');
                x0(4)            = input('Enter x4:');
                x0(5)            = input('Enter x5:');
                x0(6)            = input('Enter x6:');
                x0(7)            = input('Enter x7:');
                x0(8)            = input('Enter x8:');
                x0(9)            = input('Enter x9:');
            else
                %x0_default       = x0.*(0.1*rand(1,length(x0_default)));
                fval_min         = fval_min_base;
            end
        end
    end


end
disp('Data fit routine complete')
disp('**********************************************')
figure(fig_id+1);plot([height_data';y_test]');grid
```

```
title('DATA AFTER FITTING ROUTINE')
legend('Input data','Curve fit')
height_data            = y_test;

end
%**************************************************************************
%                           process_data_and_save
%**************************************************************************
function process_data_and_save(height_data,t_max,t_offset,fig_id,save_pname,↙
resist_type,zero_bmp_flag)
if ~exist('linearize_thickness')
    linearize_thickness = height_data;
end
disp(['Thickness modulation range = ' num2str(t_offset) ' to ' num2str(t_max+t_offset)↙
' with amplitude = ' num2str(t_max)])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           OLD METHOD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% max_indx             = find( linearize_thickness >=  t_offset );
% min_indx             = find( linearize_thickness >= (t_offset + t_max) );
% work_indx            = find( (linearize_thickness < (t_offset + t_max)) &↙
(linearize_thickness >= t_offset) );
% pow_min              = round(length(min_indx)/length(linearize_thickness)*100);
% pow_max              = round(length(max_indx)/length(linearize_thickness)*100);
% disp(['*****NOTE: Laser power min through bmp mapping = ' num2str(pow_min) '%'])
% disp(['*****NOTE: Laser power max through bmp mapping = ' num2str(pow_max) '%'])
% disp('*****WARNING: Do not adjust the DAC - Keep at min = 0% and max = 100%')
% % disp(['*****NOTE: DAC offset    = ' num2str(pow_min*10)])
% % disp(['*****NOTE: DAC amplitude = ' num2str((pow_max-pow_min)*10)])
%
% DOE_exposure         = linspace(0,1,256);
%
% max_indx             = find( linearize_thickness >=  t_offset );
% min_indx             = find( linearize_thickness >= (t_offset + t_max) );
% work_indx            = find( (linearize_thickness <= (t_offset + t_max)) &↙
(linearize_thickness >= t_offset) );
% reference_exposure   = linspace(0,1,length(work_indx));
% lin_max              = max((linearize_thickness(work_indx)-t_offset)/t_max);
% indx                 = find(DOE_exposure>lin_max);
% DOE_exposure(indx)   = repmat(lin_max,1,length(indx));
% if resist_type == 1
%     thickness_map     = (lin_max - DOE_exposure);
% else resist_type == -1
%     thickness_map     = DOE_exposure;
% end
% DOE_exposure         = interp1((linearize_thickness(work_indx)-t_offset)/t_max,↙
reference_exposure,thickness_map,'spline');
% indx                 = find(isnan(DOE_exposure));
% DOE_exposure(indx)   = repmat(0,1,length(indx));
%
```

```
% %now scale for min and max of laser power
% DOE_exposure        = pow_min/100+ DOE_exposure*(pow_max-pow_min)/100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          NEW METHOD 091911 TDM
%from process_depth_for_mulitple_test_ramps.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%work_indx           = find( (linearize_thickness <= (offset + depth)) &
(linearize_thickness >= offset) );
work_indx            = find( (linearize_thickness <= (t_offset + t_max)) &
(linearize_thickness >= t_offset) );

lin_xvec             = (-linearize_thickness(work_indx) + linearize_thickness(work_indx
(1)))/...
    (linearize_thickness(work_indx(1)) - linearize_thickness(work_indx(end)));
lin_yvec             = work_indx/length(linearize_thickness);

indx                 = find(lin_xvec>=0);
lin_xvec             = lin_xvec(indx);
lin_yvec             = lin_yvec(indx);
% DOE_exposure         = interp1(lin_xvec,lin_yvec,thickness_map,'spline');
% indx                 = find(isnan(DOE_exposure));
% DOE_exposure(indx)   = zeros(1,length(indx));

%now scale for min and max
% output_map           = uint8(round(DOE_exposure*255));
%
% if zero_bmp_flag
%     output_map(1)    = 0;
% end
%

figure(fig_id+3);plot(lin_xvec,lin_yvec);grid
xlabel('Input Value')
ylabel('Output Value')
title('Mapping Table')

% output_map(index)    = uint8(255);
    [filename, pathname] = uiputfile('*.mat', 'Enter filename to save calibration
curve',save_pname);
    if isequal(filename,0)
        disp('User selected Cancel')
        return
    else
        save_pname        = fullfile(pathname, filename);
        disp(['User selected ', save_pname])
    end

save(save_pname,'lin_xvec','lin_yvec')

end
```

D.9 Linearize

```
%linearize_BMP.m
%022421 TDM

Q_max = 255;

%parameters
fignum      = 100;
BMPfname    = "C:\Users\wuyiy\OneDrive\桌面\CGH_20-Apr-2024_141521_201_ramps.bmp";
Outfname    = "C:\Users\wuyiy\OneDrive\桌面\CGH_20-Apr-2024_141521_201_ramps_lin.bmp";
% BMPfname = "C:\Users\wuyiy\OneDrive\桌面\CGH_28-Feb-2024_023139_457_opt_ramps.bmp";
% Outfname = "C:\Users\wuyiy\OneDrive\桌面\CGH_28-Feb-2024_023139_457_opt_ramps_lin.bmp"

%get map file
S           = load("C:\Users\wuyiy\OneDrive\桌面\linear_map.mat");
lin_xvec    = S.lin_xvec;
lin_yvec    = S.lin_yvec;

%get BMP
X           = double(imread(BMPfname));
[nr, nc, np]    = size(X);
X           = squeeze(X(:,:,1));

%scale linear map
lin_xvec    = Q_max*lin_xvec;
lin_yvec    = Q_max*lin_yvec;

%linearize
Xout        = interp1(lin_xvec,lin_yvec,X(:),'spline');
indx        = find(isnan(Xout));
Xout(indx)  = zeros(1,length(indx));
Xout        = uint8(round(reshape(Xout,nr,nc)));

%view results
figure(fignum)
imagesc(X);colormap(gray);axis image
title('Original')
caxis([min(X(:)) max(X(:))])
figure(fignum+1)
imagesc(Xout);colormap(gray);axis image
title('Linearized')
caxis([min(X(:)) max(X(:))])

%Save result file
imwrite(Xout,Outfname,'bmp')

double((max(Xout(:))-min(Xout(:))))*0.0095
```

D.10 BMP to DXF Conversion

```
clc

%parameters

fname_in        = "C:\Users\wuyiy\OneDrive\桌面\CGH_20-Apr-2024_141521_201_ramps_lin.↙
bmp";
fname_out       = "C:\Users\wuyiy\OneDrive\桌面\CGH_20-Apr-2024_141521_201_ramps_lin.↙
dxf";

%%%%%%%%%%%%%%%%%%%%%REMEMBER TO CHANGE THE AU!!!%%%%%%%%%%%%%%%%%%%%%%%%%%

fignum          = 10;
fname_Header    = 'Header060321.mat';
fname_Closing   = 'Closing060321.mat';
scale           = 14;  %This scales each pixel to X microns
Layerheading    = 'L';
Layerending     = 'D0';
Polylinehead    = [{'0'} {'POLYLINE'} {'8'}];
% Example below is for a 25um linewdth
% Polylinemid     = [{'70'} {'0'} {'40'} {'25'} {'41'} {'25'} {'66'} {'1'} {'0'}↙
{'VERTEX'} {'8'}];
% need to match
Polylinemid     = [{'70'} {'0'} {'40'} {'14'} {'41'} {'14'} {'66'} {'1'} {'0'}↙
{'VERTEX'} {'8'}];
Nlevels         = 256;  %Number of grayscale levels - also number of layers

%load image file
X           = imread(fname_in,'bmp');
% X           = round(double(X)/4);  %scale to 0:63
% indx        = find(X==64);
% X(indx)     = 63*ones(1,length(indx));
[nr, nc]    = size(X);

%Setup structs and vectors
Tablevec        = [{'0'} {'SECTION'} {'2'}  {'TABLES'} {'0'} {'TABLE'} {'2'} {'LAYER'}↙
{'70'} {'255'}];
Entitiesvec     = [{'0'} {'SECTION'} {'2'} {'BLOCKS'} {'0'} {'ENDSEC'} {'0'}↙
{'SECTION'} {'2'} {'ENTITIES'}];

%load header info (saved as cell vector)
S               = load(fname_Header);
Headervec       = S.Headervec;

%load closing info (saved as cell vector);
S               = load(fname_Closing);
Closingvec      = S.Closeingvec;

%make Tables section
for ii = 1:Nlevels
    layer_name  = [Layerheading num2str(ii-1) Layerending];
```

```
    Tablevec    = [Tablevec {'0'} {'LAYER'} {'70'} {'0'} {'62'} {num2str(ii-1)} {'6'} {'CONTINUOUS'} {'2'} {layer_name}];
end
Tablevec        = [Tablevec {'0'} {'ENDTAB'} {'0'} {'ENDSEC'}];

%At this point, need to start writing DXF file, becuase the length of
%Polyline is so long
fileID          = fopen(fname_out,'w');
for ii = 1:length(Headervec)
    fprintf(fileID,[Headervec{ii} '\n']);
end
for ii = 1:length(Tablevec)
    fprintf(fileID,[Tablevec{ii} '\n']);
end
for ii = 1:length(Entitiesvec)
    fprintf(fileID,[Entitiesvec{ii} '\n']);
end

%now scan lines of the file and create polyline entities - also print
%Polyline for each row of the image
% for ii = 1:nr
%     disp(['Processing image line ' num2str(ii) ' out of ' num2str(nr)])
%     imageline   = X(ii,:);  %Scan one line
%     for jj = [0 5]%0:(Nlevels-1)   %test for each level from 1:63
%         indxjj  = find(imageline == jj);  %find which pixels have the value
%         layer_name  = [Layerheading num2str(jj) Layerending];
%         if ~isempty(indxjj) && (length(indxjj) > 1)  %process pixels with that value when more than just one pixel
%             Polylinevec     = [];
%             [position_vec, length_vec]  = process_line(indxjj);
%             for kk = 1:length(position_vec)
%                 xvalue_start    = scale*(position_vec(kk) - nc/2);
%                 xvalue_end      = scale*(position_vec(kk) + length_vec(kk) - nc/2);
%                 yvalue          = scale*(ii-nr/2);
%                 Polylinevec = [Polylinehead {layer_name} Polylinemid ...
%                     {layer_name} ...
%                     {'10'} {num2str(round(xvalue_start-scale/2))} {'20'} {num2str(round(yvalue))} ...
%                     {'0'} {'VERTEX'} {'8'} {layer_name} ...
%                     {'10'} {num2str(round(xvalue_end+scale/2))} {'20'} {num2str(round(yvalue))} ...
%                     {'0'} {'SEQEND'} ...
%                     ];
%             end
%             for kk = 1:length(Polylinevec)
%                 fprintf(fileID,[Polylinevec{kk} '\n']);
%             end
%         elseif ~isempty(indxjj) && (length(indxjj) == 1) %process pixels with that value when just one pixel
%             Polylinevec     = [];
```

```
%             xvalue      = scale*(indxjj-nc/2);
%             yvalue      = scale*(ii-nr/2);
%             Polylinevec = [Polylinehead {layer_name} Polylinemid ...
%                 {layer_name} ...
%                 {'10'} {num2str(round(xvalue-scale/2))} {'20'} {num2str(round
(yvalue))} ...
%                 {'0'} {'VERTEX'} {'8'} {layer_name} ...
%                 {'10'} {num2str(round(xvalue+scale/2))} {'20'} {num2str(round
(yvalue))} ...
%                 {'0'} {'SEQEND'} ...
%                 ];
%             for kk = 1:length(Polylinevec)
%                 fprintf(fileID,[Polylinevec{kk} '\n']);
%             end
%         end
%         %Polylinevec     = [Polylinevec ];
%     end
%
% end
total_pixels = 0;
for ii = 1:nr
    disp(['Processing image line ' num2str(ii) ' out of ' num2str(nr)])
    imageline   = X(ii,:);  %Scan one line
    for jj = 0:(Nlevels-1)   %test for each level from 0:63
        indxjj  = find(imageline == jj);  %find which pixels have the value
        layer_name  = [Layerheading num2str(jj) Layerending];
        if ~isempty(indxjj) && (length(indxjj) > 1)  %process pixels with that value
when more than just one pixel
            Polylinevec     = [];
            [position_vec, length_vec]  = process_line(indxjj);
            for kk = 1:length(position_vec)
                total_pixels = total_pixels+length(indxjj);
                xvalue_start    = scale*(position_vec(kk) - nc/2);
                xvalue_end      = scale*(position_vec(kk) + length_vec(kk) - nc/2);
                yvalue          = scale*(ii-nr/2);
               Polylinevec = [Polylinevec Polylinehead {layer_name} Polylinemid ...
                    {layer_name} ...
                    {'10'} {num2str(round(xvalue_start))} {'20'} {num2str(round
(yvalue))} ...
                    {'0'} {'VERTEX'} {'8'} {layer_name} ...
                    {'10'} {num2str(round(xvalue_end))} {'20'} {num2str(round(yvalue))}
...
                    {'0'} {'SEQEND'} ...
                    ];
            end
            for kk = 1:length(Polylinevec)
                fprintf(fileID,[Polylinevec{kk} '\n']);
            end
        elseif ~isempty(indxjj) && (length(indxjj) == 1) %process pixels with that
value when just one pixel
```

```
            Polylinevec     = [];
            total_pixels = total_pixels+length(indxjj);
            xvalue      = scale*(indxjj-nc/2)+scale;
            yvalue      = scale*(ii-nr/2);
%             Polylinevec = [Polylinehead {layer_name} Polylinemid ...
%                 {layer_name} ...
%                 {'10'} {num2str(round(xvalue-scale/2))} {'20'} {num2str(round
(yvalue))} ...
%                 {'0'} {'VERTEX'} {'8'} {layer_name} ...
%                 {'10'} {num2str(round(xvalue+scale/2))} {'20'} {num2str(round
(yvalue))} ...
%                 {'0'} {'SEQEND'} ...
%                 ];
            Polylinevec = [Polylinehead {layer_name} Polylinemid ...
                {layer_name} ...
                {'10'} {num2str(round(xvalue-scale))} {'20'} {num2str(round(yvalue))}
...
                {'0'} {'VERTEX'} {'8'} {layer_name} ...
                {'10'} {num2str(round(xvalue))} {'20'} {num2str(round(yvalue))} ...
                {'0'} {'SEQEND'} ...
                ];

            for kk = 1:length(Polylinevec)
                fprintf(fileID,[Polylinevec{kk} '\n']);
            end
        end
        %Polylinevec     = [Polylinevec ];
    end

end

%Print closing to file
for ii = 1:length(Closingvec)
    fprintf(fileID,[Closingvec{ii} '\n']);
end
fclose(fileID);


function [position_vec, length_vec]  = process_line(indxjj)

length_vec  = [];
for ii = 1:length(indxjj)
    if ii == 1
        position_vec    = indxjj(1);
        length_count    = 1;
    else
        DELTAindx   = indxjj(ii)-indxjj(ii-1);
        if DELTAindx == 1 && ii ~= length(indxjj)
            length_count    = length_count +1;
        elseif DELTAindx == 1 && ii == length(indxjj)
```

```
            length_count     = length_count +1;
            length_vec       = [length_vec length_count];
        elseif DELTAindx > 1  && ii ~= length(indxjj)
            length_vec       = [length_vec length_count];
            length_count     = 1;
            position_vec     = [position_vec indxjj(ii)];
        elseif DELTAindx > 1  && ii == length(indxjj)
            length_vec       = [length_vec length_count 1];
            position_vec     = [position_vec indxjj(ii)];
        end
    end
end
% if length(length_vec) < length(position_vec)
%     keyboard
% end
end
```

References

[1] David Engström, Anders Frank, Jan Backsten, Mattias Goksör, and Jörgen Bengtsson, "Grid-free 3D multiple spot generation with an efficient single-plane FFT-based algorithm," Opt. Express 17, 9989-10000 (2009)

[2] “Definition of Dammann Grating,” Photonics Media, photonics.com/EDU/Dammann_grating/d8153 (2024)

[3] “Hardbake, Reflow and DUV Hardening,” MicroChemicals, microchemicals.com/technical_information/photoresist_hardbake_reflow_uv_hardening.pdf (2024)

[4] H. Dammann, K. Görtler, “High-efficiency in-line multiple imaging by means of multiple phase holograms,” Optics Communications, Volume 3, Issue 5, 312-315, ISSN 0030-4018, doi.org/10.1016/0030-4018(71)90095-2 (1971)

[5] “Heidelberg MLA150 aligner for maskless photolithography,” Materials Research Laboratory UIUC, mrl.illinois.edu/facilities/equipment/heidelberg-mla150-aligner-maskless-photolithography (2024)

[6] “Holographic Grating Spectrometer,” Thorlabs, Inc., thorlabs.com/newgrouppage9.cfm?objectgroup_id=15359 (2024)

[7] “Inductively Coupled Plasma Rie Etching (ICP),” Oxford Instruments, plasma.oxinst.com/technology/icp-etching (2024)

[8] Martin, Michael, “Silicon Dioxide Plasma Etch Guide,” University of Louisville Micro/Nano Technology Center, louisville.edu/micronano/files/documents/characterizationdata/SiliconDioxidePlasmaEtchGuidev2.pdf/ (2020)

[9] “MATLAB simulation program for Physical Optics,” Figshare, figshare.com/articles/software/Matlab_Simulation_Program_for_Physical_Optics/21875958/1 (2023)

[10] “Microposit S1800 Series Photo Resists,” Shipley, kayakuam.com/wp-content/uploads/2019/09/S1800.pdf (2019)

[11] Obari, D., “What is Apple’s face ID and how does it work?,” Valent Publishing Group, makeuseof.com/apple-face-id-explained/ (2023)

[12] Polyanskiy, M., “Refractive Index of Fused Silica (Fused Quartz),” Refractiveindex.info, refractiveindex.info/?shelf=glass&book=fused_silica&page=Malitson (2024)

[13] Purvin, Kira, “Fabrication and Testing of Computer Generated Holograms Using the MLA 150,” University of Arizona, Tucson, USA. (2021)

[14] “Reactive ion etching or RIE, systems and Processes,” Corial, corial.plasmatherm.com/en/technologies/reactive-ion-etching-rie (2023)
[15] Shaun Pacheco, Jonathan F. Brand, Melissa Zaverton, Tom Milster, and Rongguang Liang, "Sensitivity analysis and optimization method for the fabrication of one-dimensional beam-splitting phase gratings," Opt. Express 23, 11771-11782 (2015)
[16] “Simulated Annealing Terminology,” The MathWorks, Inc., mathworks.com/help/gads/understanding-simulated-annealing-terminology.html (2024)
[17] Tom D. Milster, “Chapter 3 - The Gerchberg-Saxton Phase Retrieval Algorithm and Related Variations,” Optical Holography, Elsevier, 61-72 (2020)
[18] W. J. Dallas and A. W. Lohmann, "Phase Quantization in Holograms–Depth Effects," Appl. Opt. 11, 192-194 (1972)
[19] W. J. Dallas, "Phase Quantization in Holograms—a Few Illustrations," Appl. Opt. 10, 674-676 (1971)