

Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

2-1-2011

Electro-optic adaptive microlens

Dale Ewbank

Follow this and additional works at: <https://repository.rit.edu/theses>

Recommended Citation

Ewbank, Dale, "Electro-optic adaptive microlens" (2011). Thesis. Rochester Institute of Technology. Accessed from

This Dissertation is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

ELECTRO-OPTIC ADAPTIVE MICROLENS

by

DALE E. EWBANK

A DISSERTATION

Submitted in partial fulfillment of the requirements
For the degree of Doctor of Philosophy
in
Microsystems Engineering
at the
Rochester Institute of Technology

February 2011

Author: Dale E. Ewbank _____
Microsystems Engineering Program

Certified by: _____
Thomas W. Smith, Ph.D.
Professor of Chemistry and Microsystems Engineering

Approved by: _____
Bruce W. Smith, Ph.D.
Director of Microsystems Engineering Program

Certified by: _____
Harvey J. Palmer, Ph.D.
Dean, Kate Gleason College of Engineering

NOTICE OF COPYRIGHT

© 2011

Dale E. Ewbank

REPRODUCTION PERMISSION STATEMENT

Permission Granted

TITLE:

“ELECTRO-OPTIC ADAPTIVE MICROLENS”

I, Dale E. Ewbank, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: Dale E. Ewbank _____ Date: Feb 2011 _____

Electro-Optic Adaptive Microlens

By

Dale E. Ewbank

Submitted by Dale E. Ewbank in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Microsystems Engineering and accepted on behalf of the Rochester Institute of Technology by the dissertation committee.

We, the undersigned members of the Faculty of the Rochester Institute of Technology, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Microsystems Engineering.

Approved by:

Dr. Thomas W. Smith _____ Date
(Committee Chair and Dissertation Advisor)

Dr. Lynn F. Fuller _____ Date

Dr. Michael Kotlarchyk _____ Date

Dr. Bruce W. Smith _____ Date

MICROSYSTEMS ENGINEERING PROGRAM
ROCHESTER INSTITUTE OF TECHNOLOGY

February 2011

ABSTRACT

Kate Gleason College of Engineering
Rochester Institute of Technology

Degree: Doctor of Philosophy **Program:** Microsystems Engineering

Name of Candidate: Dale E. Ewbank

Title: Electro-Optic Adaptive Microlens

The goal of the present research was to demonstrate the viability of an electro-optic adaptive microlens (EOAM) system in imaging applications requiring broadband illumination in the visible region. Previous works illustrate devices that are adaptive optics but are limited in capability. Most have been designed and optimized for a particular wavelength and many of them are polarization dependent. An adaptive optical system that will function over a broadband of visible wavelengths will be useful in many imaging applications.

The tasks completed for EOAM system design and build required understanding and implementation of the imaging theory, the materials' properties, the control voltages, the fabrication processes, and finally understanding and implementation of the imaging theory for testing. Single cell transmission devices were used for initial characterization of the polymer-dispersed liquid crystal (PDLC) process. Three iterations of the EOAM devices with PDLC were built on silicon wafers and 26 devices were optically tested. The new chemical mechanical planarization process was implemented for the second and third builds. For optical device testing the phase shift was extracted using a newly developed method for blind phase extraction.

The development of a design model for the EOAM system and validating it with the images formed by a real electro-optic adaptive microlens system has provided the knowledge base needed for implementation of adaptive electro-optic lenses for the visible region, and a process which can be used for further improvement of the microsystem. The model parameters can be adjusted for new electro-optic materials that may become available that do not have the limitations of PDLC.

Abstract Approval: Committee Chair: Thomas W. Smith

Program Director: Bruce W. Smith

Dean, KGCOE: Harvey J. Palmer

ACKNOWLEDGMENTS

This work is a compilation of efforts of numerous people from many different disciplines. While completing this dissertation, I am fortunate to have had the opportunity to collaborate with others and to enjoy the discovery of new knowledge. Thank you to all who have helped me with this task.

The author wishes to thank Dr. Thomas W. Smith for his time, ideas, and continuous encouragement during this dissertation process. It was an honor for me to have on my committee Dr. Lynn F. Fuller, Dr. Michael Kotlarchyk, and Dr. Bruce W. Smith.

I am grateful to the following organizations for support: KGCOE FEAD (2005), Microsystems Engineering, Electrical and Microelectronic Engineering, Imaging Science, Semiconductor and Microsystems Fabrication Laboratory.

I am indebted to my many colleagues for their help and support: Dr. Robert Pearson, Dr. Santosh Kurinec, Dr. Michael Jackson, Dr. Karl Hirschman, Dr. Sean Rommel, Dr. Surendra Gupta, Dr. Christopher Hoople, Dr. Mustafa Abushagur, Dr. Zoran Ninkov, Dr. Jonathan Arney, Dr. Roger Easton, Robert MacIntrye, Robert Kraynik, Scott Blondell, Thomas Grimsley, David Yackoff, Bruce Tolleson, John Nash, Richard Battaglia, Sean O'Brien, Dr. Alan Raisanen, Dan Brown, Deoram Persaud, Ivan Puchades, Jianming Zhou, Frank Cropanese, Andrew Estroff, Neal Lafferty, Germain Fenger, Christopher Shea, and the numerous other students who helped and encouraged me.

I am especially thankful to Nitin Nampalli for his optimization and characterization of the CMP process, and to Patrick Whiting for his fabrication processing and electrical simulations.

TABLE OF CONTENTS

Abstract	iv
List of Figures	viii
List of Tables	xi
I. Dissertation Statement	1
II. Survey of Related Work	3
III. Results and discussion	11
A. Imaging Theory	11
1.0 Fresnel Propagation	11
2.0 Simulation of the EOAM system	18
2.1 Function block diagram	19
2.2 Assumptions for using propagation model for the EOAM	22
B. Electro-optic Adaptive Microlens materials in single cell device	26
C. Control voltages for the arrayed pixel device	31
D. Fabrication process for the arrayed pixel device	34
E. Device testing	45
1. Dual beam interferometer	45
2. Single beam interferometer	49
a. Theory	49
b. Reflection simulation of film stack	52
c. Imaging theory for phase extraction	55
d. Assumptions and algorithm	58
e. Simulations	63

f. Summary of Phase extraction from devices.....	67
F. Summary of EOAM Results.....	72
IV. Conclusions.....	75
REFERENCES	77
Appendix A: setupworkspace160_3.m, lensf500bit_3.m, arrayfillbit_160.m, fPropfocal_160.m, and address_fbit.m	
Appendix B: datain_rerun_singlefile.m	
Appendix C: Run Sheet for EOAM v1.5b	
Appendix D: EOAM_Process Rev1_5b.PPT	
Appendix E: g5_nm_PDLC.fig and g5_nm_PDLC.m	
Appendix F: xu_2007_ph_ext_05182010_data_p.m	

List of Figures

Figure 1: This is the conceptual design and operation of the one-dimensional LC reflection mode beam steerer. Reprinted with permission from [13]	6
Figure 2: Fabrication of an inhomogeneous PDLC using a patterned photomask. Reprinted with permission from [14].....	7
Figure 3: Diffraction properties at $\lambda = 514$ nm of a prism grating made of inhomogeneous PDLC. Reprinted with permission from [14]	7
Figure 4: Method for fabricating a PDLC Fresnel lens. Reprinted with permission from [15]. Copyright 2003, American Institute of Physics.	8
Figure 5: Experimental setup for subjective feedback loop to improve visual acuity and determine aberrations of the human eye. Reprinted with permission from [2]	9
Figure 6: An adaptive LC lens fabricated for experiment and 3D model of a wireless implantable LC corrector lens are shown. Reprinted with permission from [1] ...	10
Figure 7: Geometry for aperture Σ propagating to new plane	13
Figure 8: Fresnel Wave Propagation System for Electro-optic Adaptive Microlens.	20
Figure 9: Input mask for EOAM system. Mask represents group of incoherent point sources with random phase and transmission of 1.0 for all wavelengths	24
Figure 10: Sample output image for EOAM system. Simulated wavelengths are 587, 486, and 656 nm.....	25
Figure 11: Chemical structure for E48 liquid crystal.....	26
Figure 12: Chemical structure for NOA81	27
Figure 13: PDLC at 24% by weight in NOA81 polymerized at different intensities. Sample transmission data includes losses due to two ITO coated glass slides.....	28
Figure 14: Image of single cell device made from E48/NOA81 utilizing patterned SU-8 resist as the spacer.....	28
Figure 15: Diagram of dual beam interferometer test system used for single cell devices	29
Figure 16: Phase shift versus voltage for single cell device at three wavelengths	30
Figure 17: Comparison of phase shift with illumination of two different polarizations ...	31
Figure 18: Diagram of voltage distribution board for EAOM device	33

Figure 19: Sub array of pixels for EOAM device with 16 by 16 pixels	34
Figure 20: Addresses for 8 by 8 array of a single quadrant with 160 micrometer pixels ..	35
Figure 21: Metal 1 of EOAM.....	36
Figure 22: Via 1 of EOAM.....	37
Figure 23: Metal 2 of EOAM.....	38
Figure 24: Spacer layer for EOAM.....	39
Figure 25: Layer stack for EOAM device.....	40
Figure 26: Examples of surface roughness before and after the CMP process [26] on device pixels.....	41
Figure 27: Hundred millimeter device wafer after all lithography steps completed	42
Figure 28: Finished EOAM device	48
Figure 29: Dual beam interferometer test system with device and output image on card.	46
Figure 30: Device with probes in dual beam interferometer test system.....	46
Figure 31: Interferogram of device from dual beam interferometer test system	47
Figure 32: Interferogram of device from dual beam interferometer test system	47
Figure 33: Interferogram of device from dual beam interferometer test system	48
Figure 34: Plot showing relative phase variability from dual beam interferometer test system analysis.....	48
Figure 35: Single arm interferometer system diagram	50
Figure 36: Interferogram from EOAM device active area and surround at 0 volts	51
Figure 37: Interferogram from EOAM device active area and surround at 200 volts	51
Figure 38: EOAM device with non-active area masked off by black tape.....	52
Figure 39: Modeled reflectance for EOAM device (entire film stack)	54
Figure 40: Modeled reflectance for top 4 layers of EOAM device	55
Figure 41: Flow diagram of algorithm showing its process and validation steps.....	63

Figure 42: Phase extraction simulated with varying $A_r > A_o$ maximum	65
Figure 43: Phase with errors less than $2\pi/100$ comparing input and output A_r/A_o ratio	67
Figure 44: Layers of materials used in the reflective micro-device.....	68
Figure 45: Diagram of single arm interferometer system used to collect interferograms .	68
Figure 46: Interferograms of PDLC device at 3 volts (left) and 240 volts (right).....	69
Figure 47: Sample phase extraction from interferograms of electro-optic adaptive microlens devices.....	71
Figure 48: Delta phase versus electro-optic material thickness for Δn	74

List of Tables

None

I. Dissertation Statement

The major limitations of diffractive and adaptive imaging systems presently in use are limited wavelength bandwidth and polarization dependence of the illumination. Diffractive optical elements (DOEs) are efficient in beam steering, phase modulation, image formation and scanning over a limited bandwidth due to the dispersion properties of the materials of the DOE. The adaptation of liquid crystal display (LCD) technologies to phase arrayed imaging systems has been limited due to the polarization dependence of LCDs. The use of adaptive electro-optic type devices for correction of human vision has been studied [1-4] and patented [5]; however, the implementation of corrective lens devices has not reached the consumer.

The overall goal of this research project is to demonstrate the viability of an electro-optic adaptive microlens (EOAM) system that does not have polarization dependence in imaging applications that require broadband illumination in the visible region.

The specific objective is to evaluate imaging quality as a function of the pixel array design, properties of the EOAM materials, and the applied field for each pixel in the array. The imaging quality will be evaluated by comparison of the EOAM system image to that of a simulated image. The cell array design parameters include pixel size, pixel pitch, and array size, all of which ultimately define the size of the EOAM device. The EOAM materials properties are divided into several categories: mechanical, chemical, electrical, and optical. The mechanical and chemical properties relate mainly to the fabrication steps involved in building the EOAM device. The optical properties are also relevant during fabrication as the patterning of the device is done using lithography.

The electrical and optical properties are the major contributors to the use and successful application of the EOAM device in imaging. The applied field for each pixel will set the relative phase shift for that pixel. The EOAM system is adaptive and can be used as various type of diffractive elements by appropriate choice of the pixel array design, properties of the EOAM materials, and the applied field for each pixel in the array.

The parameter space used in modeling and fabrication of the EOAM system lends itself to applications in the visible region. While the electro-optic material used for this research is not appropriate for consumer production; the knowledge developed for modeling and fabrication is applicable to novel electro-optic materials as they become available.

The polymer-dispersed liquid crystal (PDLC) material use in this work has limited change in refractive index. The drive voltage and PDLC thickness combination limit optical phase change to approximately 2 radians. This limit does not allow operation of the EOAM at 2 levels ($0, \pi$). Thus the lensing capabilities were not evaluated for a pixilated device.

II. Survey of Related Work

With the slogan "you press the button, we do the rest," George Eastman put the first simple camera into the hands of a world of consumers in 1888. In so doing, he made a cumbersome and complicated process easy to use and accessible to nearly everyone [6].

The proliferation of optical devices over the last 120 years has grown tremendously, largely on this same premise. There has always been a need and drive for creating new optical instrumentation for the scientific, academic and industrial communities; however, the needs and desires of the "consumers" are by far the largest driving force for creation of new optical devices.

As an example, since Arthur L. Schawlow and Charles H. Townes published their technical paper on the principles of the laser in 1958, the device has been put to work in a vast range of applications and has assumed many forms. Today, lasers are used in a wide range of applications in medicine, manufacturing, the construction industry, surveying, consumer electronics, scientific instrumentation, and military systems. Literally billions of lasers are at work today, ranging in size from tiny semiconductor devices no bigger than a grain of salt to high-power instruments as large as an average living room [7].

Consumer optical devices are all about making cumbersome and complicated processes easy to use and accessible to everyone. This has been done for digital cameras, video recorders, CD and DVD players, and optical storage for computers. As the old adage says "a picture is worth a thousand words," and the consumer world is full of images.

Over the past 40 years there have been two technologies that have developed and are now interacting. Liquid crystal technology has been dominant in the development of

electronic displays [8]; while optical phased array technology [9] has moved from mechanical steering to coherent optical sensor systems.

The technological development of liquid crystal displays (LCDs) began in 1964 with the discovery of guest-host mode and dynamic scattering mode by Heilmeyer of RCA Laboratories. Heilmeyer conceived the idea of manufacturing wall-sized flat-panel color televisions. But his idea did not become a reality until 1991. Due to materials properties and power requirements, until 1988 LCDs were limited to niche applications of small-size displays such as digital watches and pocket calculators. The development of twisted nematic (TN) mode, super TN mode, and liquid crystals that can operate at room temperature broadened the number of applications. Also the development of an amorphous silicon field-effect transistor allowed for addressing and control of the LCDs. Using a thin film transistor array in 1988, Nagayasu [10] of Sharp Corporation demonstrated an active-matrix full color full-motion 14 inch display. This set the new and ever advancing standard for the notebook computer industry.

These LCDs were developed and optimized for display devices, in which the major requirements were optical transmission and low power consumption. The optical properties of the LC also allow for modification of polarization of the illumination and for phase change. The polarization effects were exploited by Schadt and Helfich [11] and the utilization of two polarizers and surface alignment of the TN mode LC is the basis for most display manufacturing throughout the world.

In 1994 beam steering of visible light was reported using a LC television panel as a phased array [12]. LCDs are usually configured to modulate intensity, however when the polarizers are removed the change of phase of the illumination can be utilized. This

LC system was configured such that the display pixels created a discrete blazed-grating phase ramp across the aperture. The steering efficiency and deflection angle were limited by the large pixel size and the limited available phase modulation of 1.3π . This type of device is also polarization dependent.

Numerous other systems have been developed and many improvements have been incorporated. One dimensional phase modulation devices have been incorporated into systems for two dimensional steering. These systems were designed and optimized to solve a particular imaging problem (beam splitting or steering, scanning, focusing, and/or correction of phase aberrations). The imaging problems must be well defined and generally are severely constrained by their input and output environments. The system design is usually monochromatic and thus constrained to an operating wavelength of radiation and a very small bandwidth around that wavelength. The device size (pixel size and number of pixels) is dictated by the required numerical aperture of the system.

A phase profile is imparted on the optical wavefront when it is transmitted or reflected from the device. As this phase modified wavefront is propagated through the system it converges or diverges to form an image. The refractive index changes of the LC under an applied field allow for effective changes to the optical path length (OPL). The change in OPL results in the change of phase for that region of the wavefront.

Figure 1 shows the conceptual design for a one dimensional beam steering device developed in 1996 at a wavelength of $10.6 \mu\text{m}$ with a LC phase array [13]. The device works in reflective mode and utilizes phase wrapping of 2π . It is polarization dependent and the array has four cells.

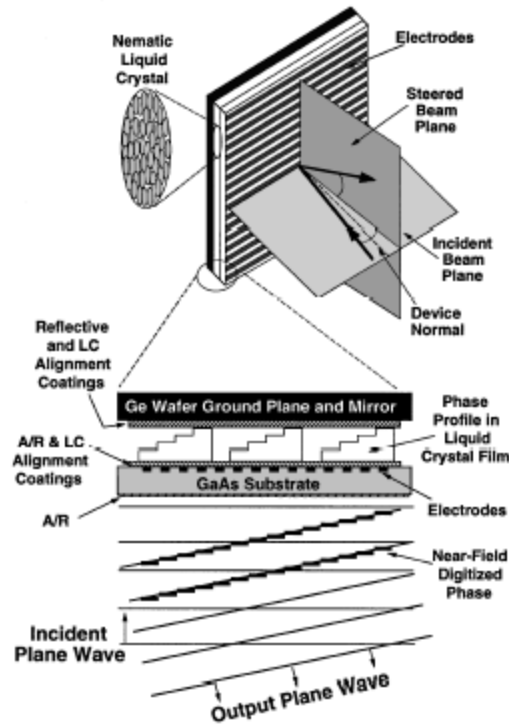


Figure 1: This is the conceptual design and operation of the one-dimensional LC reflection mode beam steerer. Reprinted with permission from [13].

Recent work has been done using polymer-dispersed liquid crystal (PDLC) materials to create devices [14, 15]. The inhomogeneous nanoscale droplets of PDLC were obtained by exposing the LC/monomer with ultraviolet (UV) radiation through a patterned photomask as shown in Figure 2. The intensity variation during exposure of the PDLC results in a gradient of droplet sizes in the film, and the relative refractive index change under an applied electric field is a function of the droplet size. This gradient refractive index nanoscale (GRIN) PDLC is highly transparent in the visible wavelengths and has been used to create prism gratings, as well as positive, negative, and Fresnel lens. The GRIN PDLC devices are broadband, independent of light polarization, and simple to fabricate, however, the required driving voltage is higher than $100 V_{\text{rms}}$ and response time is 0.080-0.200 milliseconds.

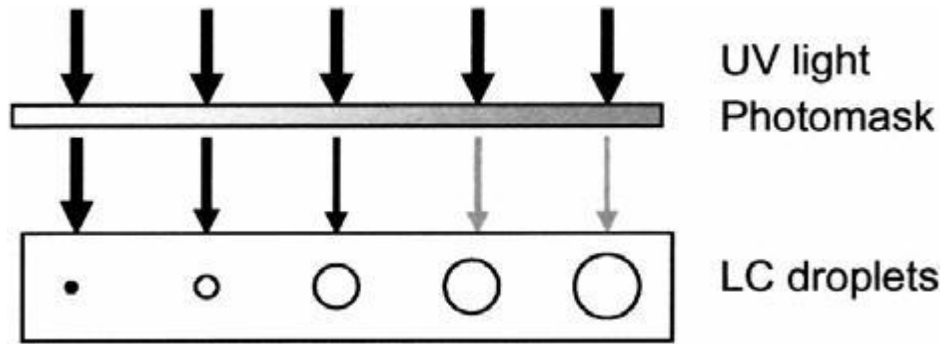


Figure 2: Fabrication of an inhomogeneous PDLC using a patterned photomask. Reprinted with permission from [14].

The optical results for a GRIN PDLC device are shown in Figure 3. The prism grating is formed by the gradient of refractive index due to the patterned polymerization resulting in control of the LC droplet sizes. The grating is “on” with no field applied and the applied field of 100 volts causes the LC droplets to align and cancel the index gradient.

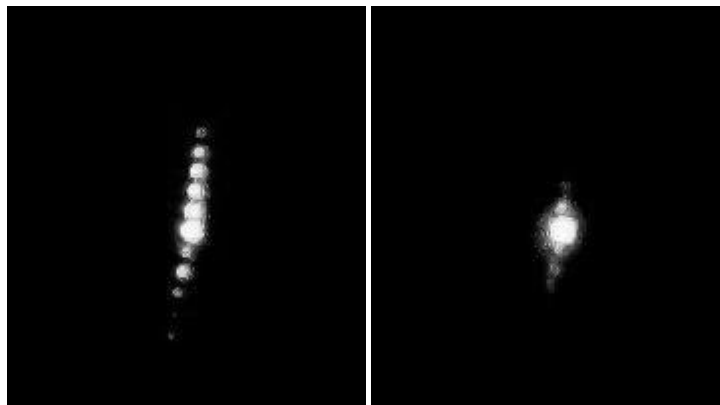


Figure 3: Diffraction properties at $\lambda = 514$ nm of a prism grating made of inhomogeneous PDLC. Reprinted with permission from [14].

A Fresnel lens was also fabricated using the GRIN PDLC as shown in Figure 4. The lens is patterned to control droplet size and thus phase in 80 zones. As with the grating above, the entire device is controlled with a single applied field. These devices are adaptable but only over their limited range of design.

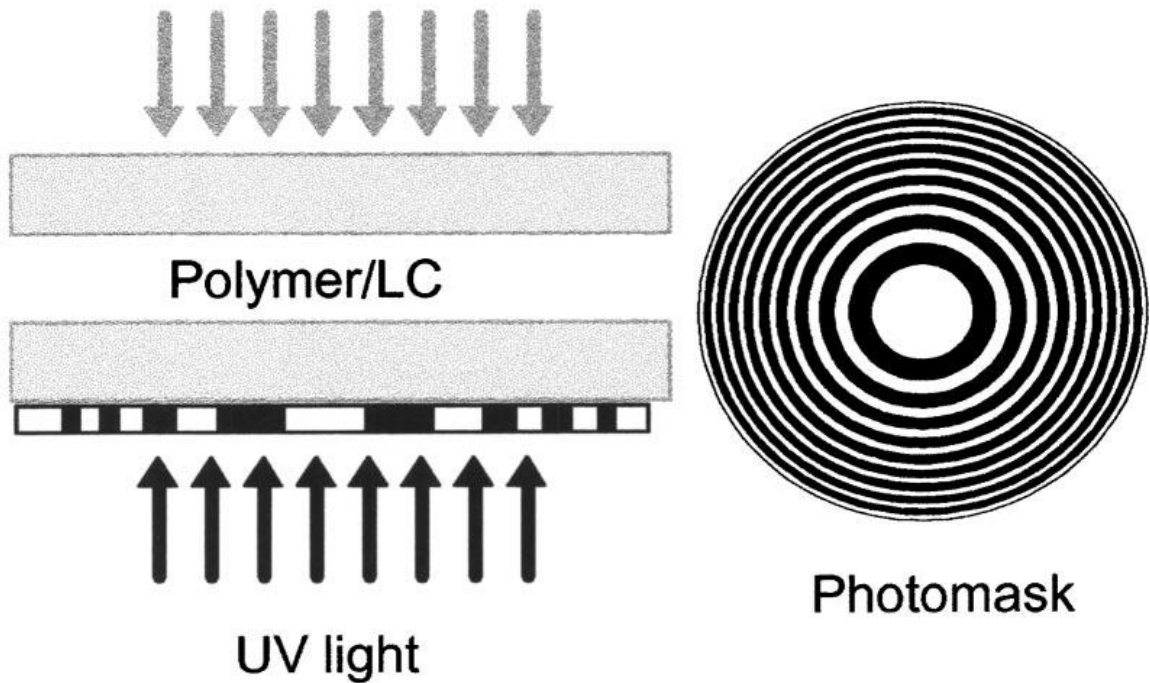


Figure 4: Method for fabricating a PDLC Fresnel lens. Reprinted with permission from [15]. Copyright 2003, American Institute of Physics.

Systems for phase modulation have also been demonstrated by using LC on silicon technology [16] and spatial light modulators (SLM) built with optically addressable LC cells [17]. These systems have shown excellent results in wavefront phase modulation and have two-dimensional array control; however, they are polarization dependent. And because the LC system is based on light scatter they suffer from limited efficiency of light transfer.

One of the applications for wavefront phase modulation is correction for human vision. An adaptive optics phoropter system has been demonstrated utilizing optically addressable LC SLM [18]. The system is used to measure the wavefront errors that occur due to the structure of the human eye. The adaptive optics allow for correction of the lower-order aberrations of the eye (defocus and astigmatism) as can be done using a corrective lens. Additionally, the adaptive wavefront is capable of dealing with the high-

order aberrations such as spherical aberration and coma, leading to near diffraction-limited image quality at the retina.

Figure 5 shows the test bench design for a correction system with human feedback for control of a deformable mirror for wavefront control [2]. The system is designed to correct wavefront errors that limit human vision and to establish the correction values needed for proper wavefront control. Vdovin has also done work in the field of applying these corrections by building adaptive lens [1]. As seen in Figure 6, a 5 mm aperture LC adaptive lens was fabricated. The lens is addressed by a single applied field and only focal length can be controlled. This LC system is also polarization dependent.

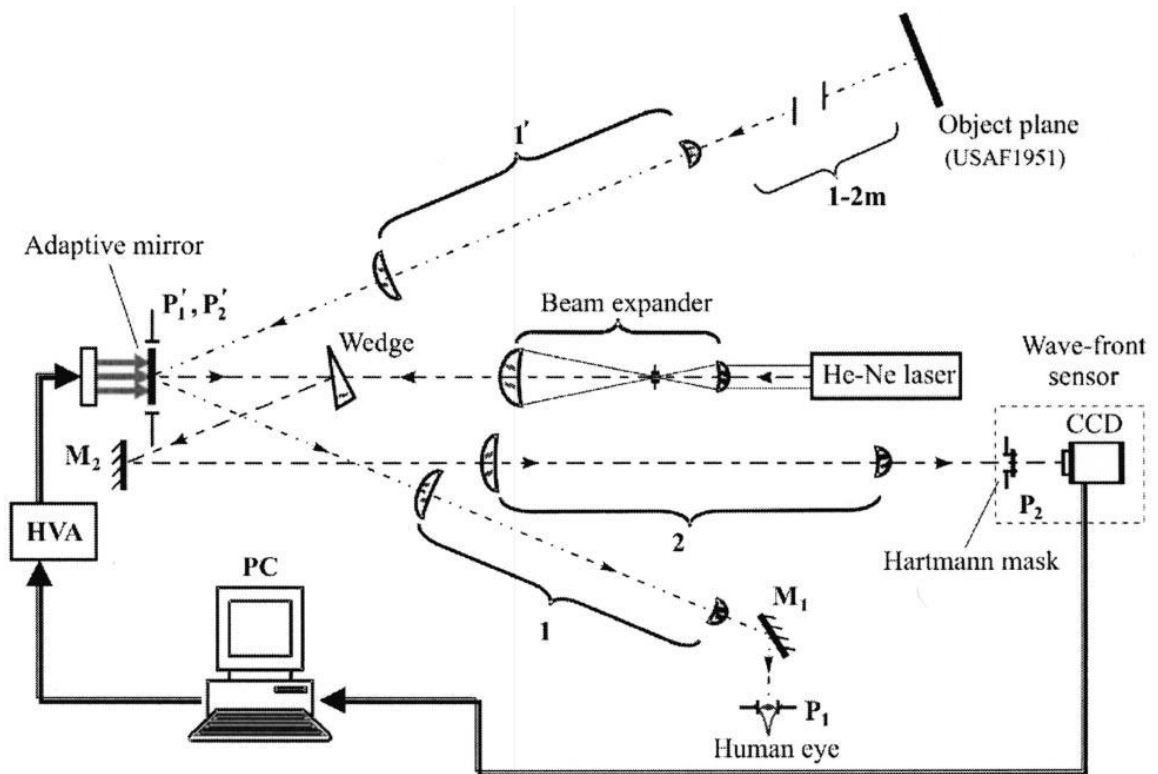


Figure 5: Experimental setup for subjective feedback loop to improve visual acuity and determine aberrations of the human eye. Reprinted with permission from [2].

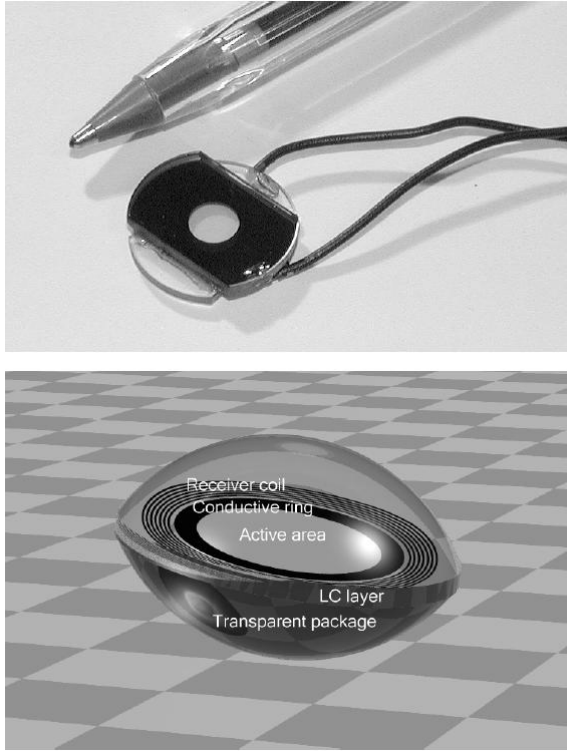


Figure 6: An adaptive LC lens fabricated for experiment and 3D model of a wireless implantable LC corrector lens are shown. Reprinted with permission from [1].

The previous works illustrate devices that are adaptive optics but are limited in capability. Most have been designed and optimized for a particular wavelength and many of them are polarization dependent. An adaptive optical system that will function over a broadband of visible wavelengths and be polarization independent will be useful in many imaging applications.

III. Results and discussion

The five major topics of the solution are understanding and implementation of the imaging theory for design, the materials' properties, the control voltages for the device, the fabrication process, and finally understanding and implementation of the imaging theory for testing. Each of these topics will be discussed in this section.

A. Imaging Theory

An essential element of this research project is the design, modeling, and fabrication of an EOAM system for use in the visible wavelengths of light. An EOAM that will function over a broad range of applications can reduce the cost of production and ultimately reduce the cost of ownership for the system.

1.0 Fresnel Propagation

The optical modeling for the EOAM system is based on Fresnel propagation [19-22]. The device design was simulated and the imaging system was modeled and compared to the desired output (image). In an iterative process the design can be changed, simulated, remodeled and compared to allow for optimization of the EOAM. The design changes can be driven by various optimization schemes.

Viewing wave propagation phenomena as a system allows for valid approximations over a wide class of input field distributions and optical elements. The concept of the intensity of a wave field and the Huygens-Fresnel principle are well suited for approximation in image formation. The first wave theory for light expressed by Christian Huygens in 1678 was that if each point on a wavefront is considered as a point source radiating spherical wavefronts, then a later wavefront can be found by constructing an envelope of the secondary wavelets.

The response of a detector is a function of the distribution of the intensity in the image. Thus it is important to relate the intensity to the complex field which makes up the image.

In 3-dimensional space the second order partial differential wave equation is

$$\nabla^2 f(\vec{r}, t) = \frac{1}{c^2} \frac{\partial^2 f(\vec{r}, t)}{\partial t^2} \quad (3.1.1)$$

and the generalized harmonic wave is

$$f(\vec{r}, t) = E(\vec{r}) \cos(S(\vec{r}) - \omega t) \quad (3.1.2)$$

where $\vec{r} = \hat{i}x + \hat{j}y + \hat{k}z$ represents a position vector of a point in space. At any fixed time, the surfaces for which $S(\vec{r})$ equals a constant are called wavefronts. When $E(\vec{r})$, the amplitude of the wave, is a constant over the wavefront, the wave is homogeneous.

The above generalized harmonic wave can be expressed in complex form as

$$f(\vec{r}, t) = \Psi(\vec{r}) e^{-i\omega t} \quad (3.1.3)$$

where

$$\Psi(\vec{r}, t) = E(\vec{r}) e^{i[S(\vec{r}) + \varphi]} \quad (3.1.4)$$

with φ equal to an initial arbitrary phase.

This is useful when substituting $f(\vec{r}, t)$ into (3.1.1) resulting in

$$\begin{aligned} \frac{\partial^2 f(\vec{r}, t)}{\partial t^2} &= -\omega^2 \Psi(\vec{r}) e^{-i\omega t} \\ \nabla^2 f(\vec{r}, t) &= e^{-i\omega t} \nabla^2 \Psi(\vec{r}) \\ e^{-i\omega t} \nabla^2 \Psi(\vec{r}) &= \frac{-\omega^2}{c^2} \Psi(\vec{r}) e^{-i\omega t} \\ (\nabla^2 + k^2) \Psi(\vec{r}) &= 0 \end{aligned} \quad (3.1.5)$$

which is the Helmholtz Equation. If interested in the spatial properties, but not the temporal, solutions to the Helmholtz Equation are sufficient to represent the wave.

The Huygens-Fresnel principle can be stated in rectangular coordinates as

$$\psi(x, y; z = z_1) = \frac{1}{i\lambda} \iint_{\Sigma} \psi(x_0, y_0; z = z_0) \frac{\exp(ikr_{01})}{r_{01}} \cos(\theta) dx_0 dy_0 \quad (3.1.6)$$

where the angle θ between the outward normal and the vector \vec{r}_{01} pointing from P_1 to P_0 as shown in Figure 7. $\psi(x, y; z = z_1)$ represents the field at the plane of z_1 having

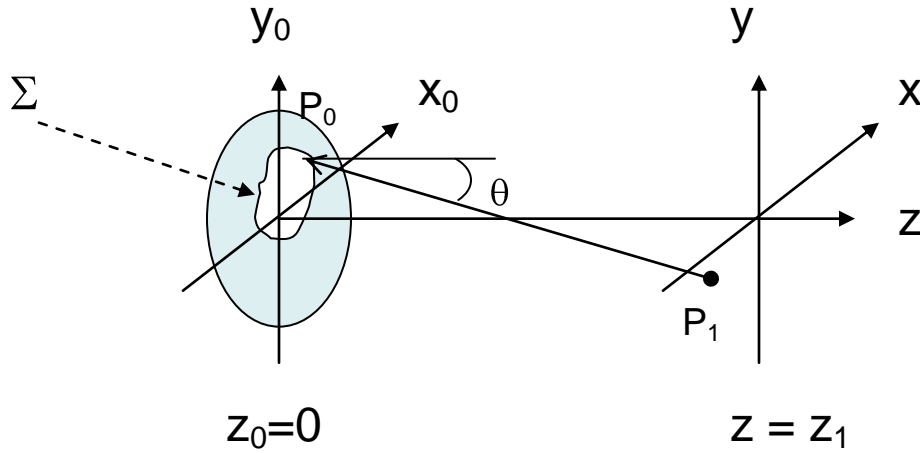


Figure 7: Geometry for aperture Σ propagating to new plane.

propagated from the plane at z_0 . The value of $\cos(\theta) = \frac{z_1}{r_{01}}$ and (3.1.6) can be rewritten as

$$\psi(x, y; z = z_1) = \frac{z_1}{i\lambda} \iint_{\Sigma} \psi(x_0, y_0; z = z_0) \frac{\exp(ikr_{01})}{r_{01}^2} dx_0 dy_0 \quad (3.1.7)$$

where the vector \vec{r}_{01} distance is given by

$$\begin{aligned} r_{01} &= \sqrt{z_1^2 + (x - x_0)^2 + (y - y_0)^2} \\ &= z_1 \left[1 + \frac{(x - x_0)^2 + (y - y_0)^2}{z_1^2} \right]^{1/2}. \end{aligned} \quad (3.1.8)$$

Huygens-Fresnel principle has only two approximations: one is the approximation inherent in scalar theory and the second is the assumption that $r_{01} \gg \lambda$ as the observation plane is many wavelengths from the aperture.

The distance r_{01} can be approximated by making use of the binomial expansion for the square root. For $b \leq 1$ the number of terms needed in the expansion

$$\sqrt{1+b} = 1 + \frac{1}{2}b - \frac{1}{8}b^2 + \dots, \quad (3.1.9)$$

for accuracy depends on the magnitude of b . Applying the expansion to (1.8) yields

$$\begin{aligned} r_{01} &\approx z_1 \left[1 + \frac{1}{2} \left(\frac{(x-x_0)^2 + (y-y_0)^2}{z_1^2} \right) \right] \\ &\approx z_1 + \left(\frac{(x-x_0)^2 + (y-y_0)^2}{2z_1} \right) \end{aligned} \quad (3.1.10)$$

by retaining the first two terms. When substituting (3.1.10) into (3.1.7) the error of the value for r_{01} squared in the denominator is small provided $z_1 \gg (x-x_0)^2 + (y-y_0)^2$, which is the case in the paraxial region. However, the r_{01} in the exponent is multiplied by a large $k = \frac{2\pi}{\lambda}$, and phase changes of small fractions of a radian change the value of the exponential significantly. Both terms in the binomial approximation must be kept in the exponent. The expression for the field by substituting (3.1.10) into (3.1.7) becomes

$$\psi(x, y) = \frac{z_1}{i\lambda} \iint_{\Sigma} \psi(x_0, y_0) \frac{\exp\left(ik \left(z_1 + \left(\frac{(x-x_0)^2 + (y-y_0)^2}{2z_1} \right) \right)\right)}{z_1^2} dx_0 dy_0.$$

By rearranging and incorporating the finite limits of the aperture Σ in the definition of $\psi(x_0, y_0)$ the resulting equation is

$$\psi(x, y) = \frac{\exp(ikz_1)}{i\lambda z_1} \int \int_{-\infty}^{\infty} \psi(x_0, y_0) \exp\left(i \frac{k}{2z_1} \left((x-x_0)^2 + (y-y_0)^2\right)\right) dx_0 dy_0. \quad (3.1.11)$$

The field at any plane z_1 described by (3.1.11) can be seen as a convolution of the form

$$\psi(x, y) = \int \int_{-\infty}^{\infty} \psi(x_0, y_0) h(x-x_0, y-y_0) dx_0 dy_0 \quad (3.1.12)$$

with the convolution kernel as

$$h(x, y, z_1) = \frac{\exp(ikz_1)}{i\lambda z_1} \exp\left(i \frac{k}{2z_1} (x^2 + y^2)\right). \quad (3.1.13)$$

The expression for $h(x, y, z_1)$ represents a diverging spherical wave and quadratic phase approximation to the wave for position values of z_1 . The convolution with (3.1.13) is the propagation of the field from the aperture $\psi(x_0, y_0)$ at plane $z=0$ to the field at $\psi(x, y)$ for the plane at z_1 .

Maxwell's Equations lead to the properties of light [19-22]: its wave nature, that it is a transverse wave, and the relationship between the \vec{E} electric and \vec{B} magnetic fields.

Assume light propagating in a medium that has the following properties:

- Uniform: ϵ , permittivity (dielectric constant,) and μ , permeability, have constant value at all points
- Isotropic: ϵ and μ do not depend on direction of propagation
- Nonconducting: $\sigma = 0$, conductivity, and thus $\vec{J} = 0$, current density
- Free of "free charge": $\rho = 0$, charge density
- Nondispersive: ϵ and μ are independent of frequency.

Then Maxwell's Equations are:

$$\nabla \cdot \vec{E} = 0 \quad (3.1.14)$$

$$\nabla \cdot \vec{B} = 0 \quad (3.1.15)$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \quad (3.1.16)$$

$$\nabla \times \frac{\vec{B}}{\mu} = -\frac{\partial \epsilon \vec{E}}{\partial t} + \vec{J} \quad (3.1.17)$$

Evaluating Maxwell's Equation in a medium leads to the following two equations

$$\nabla^2 \vec{E} = \mu \epsilon \frac{\partial^2 \vec{E}}{\partial t^2} \quad (3.1.18)$$

$$\nabla^2 \vec{B} = \mu \epsilon \frac{\partial^2 \vec{B}}{\partial t^2}, \quad (3.1.19)$$

which are coupled transverse waves. The electric and magnetic field are also solutions to the Helmholtz and 3-dimension second order partial differential wave equation for vacuum when

$$c = \frac{1}{\sqrt{\mu_0 \epsilon_0}}. \quad (3.1.20)$$

For materials where v , velocity, is less than c , velocity in vacuum, the material is characterized by its index of refraction,

$$n = \frac{c}{v} = \sqrt{\frac{\mu \epsilon}{\mu_0 \epsilon_0}}. \quad (3.1.21)$$

The intensity of the field is related to the flow of energy for the coupled electric and magnetic fields as described by the Poynting Vector,

$$\vec{S} = \vec{E} \times \frac{\vec{B}}{\mu}. \quad (3.1.22)$$

The Poynting vector cannot be detected at the very high frequencies associated with light, so what is detected is the temporal average of \vec{S} taken as an average over time, T , determined by the detector time response. The time average of \vec{S} is the flux density in units of $[\text{W}/\text{m}^2]$ and is called intensity of the light wave,

$$I = \left| \langle \vec{S} \rangle_T \right| = \frac{1}{T} \int_{t_0}^{t_0+T} \left(\vec{E} \times \frac{\vec{B}}{\mu} \right) dt. \quad (3.1.23)$$

For an electric field represented by $\vec{E} = \vec{E}_0 \cos(\omega t - \vec{k} \cdot \vec{r} + \phi)$ and using $\frac{\vec{B}}{\mu} = \frac{n}{\mu c} \frac{\vec{k} \times \vec{E}}{k}$ the expression for intensity can be reduced to

$$I = \left| \langle \vec{S} \rangle_T \right| = \frac{1}{T} \int_{t_0}^{t_0+T} \vec{A} \cos^2(\omega t - \vec{k} \cdot \vec{r} + \phi) dt \quad (3.1.24)$$

where $\vec{A} = \frac{n}{\mu c} |\vec{E}_0|^2 \frac{\vec{k}}{k}$. Then for \vec{k} independent of time and $\omega t \gg 1$,

$$I = |\vec{A}| \frac{1}{\omega T} \int_{\omega t_0}^{(\omega t_0 + T)\omega} \cos^2(\omega t - \vec{k} \cdot \vec{r} + \phi) d(\omega t). \quad (3.1.25)$$

Knowing that for large P the integral $\frac{1}{P} \int_{x_0}^{x_0+P} \cos^2 \theta d\theta = 1/2$, results in

$$I = \frac{1}{2} \frac{n}{\mu c} |\vec{E}_0|^2. \quad (3.1.26)$$

The intensity of the field is proportional to the electric field amplitude squared.

In actual practice the electric field at the object plane must be broken down into constituent parts for modeling. The above Fresnel propagation is based on monochromatic illumination and incorporating the finite limits of the aperture, Σ , in the

definition of $\psi(x_0, y_0)$ for (3.1.11), also assumes that the aperture can be adequately described.

For broadband illumination in the visible region the propagation can be simulated at multiple wavelengths. Then a summation of the multiple wavelength aerial images approximates the actual image well. The difficulty in this scenario is the estimation of temporal coherence.

To adequately describe the illumination field at aperture, Σ , for the case when the field is spatially coherent is relatively easy. An arbitrary phase ϕ can be assigned as in (3.1.4). However, for the spatially incoherent field this is not possible. In the spatially incoherent field case it is useful to evaluate the EOAM system as if the illumination field is a point source, and to use the aerial image of that point source as the system response. The system response from the point source is valid for spatial coherence; however, the effects of temporal coherence may introduce errors. The image from the aperture, Σ , can then be approximated by proper sizing (magnification) and convolution with the system response as Fresnel propagation is linear and shift invariant [20].

2.0 Simulation of the EOAM system

The purpose of the system is to create an aerial image for a finite amount of time that can be captured by another system for viewing, propagation or as a latent image. A wavefront (electro-magnetic field) and a mask are the inputs to the system. The output is an aerial image that has characteristics unique to the system inputs. The desired image is a result of the transfer of the wavefront and mask as well as the interactions of the various system functions. The mask may be designed such that it resembles the output aerial

image or it may be designed to contain information that changes the wavefront unique to the system.

The input wavefront is allowed to enter the system for a finite time by the exposure control unit. The mask in the filter interface then modulates this wavefront. The resulting modified wavefront is then propagated a distance z_1 . The propagation of the wavefront results in a redistribution of the energy in the wavefront. The EOAM function then collects the energy and modifies the wavefront. The wavefront leaving the EOAM is then propagated a distance z_2 . The wavefront output is an electro-magnetic field that has an energy distribution that can be viewed, propagated, or captured.

2.1 Function block diagram

The function block diagram in Figure 8 illustrates the system relationships used in the simulation and for the physical EOAM device. Each of the components (inputs, output, and functions) is described in this section.

I_1 – Wavefront $E(x, y, z, t, \lambda)$

The wavefront is an electro-magnetic wave that is described in four-dimensional space. The wavefront is known by its electric component, which is a complex vector field, $E(x, y, z, t, \lambda)$, as a function of three-dimensional space and time. The wave must follow Maxwell's Equations as it propagates and interacts with space and matter.

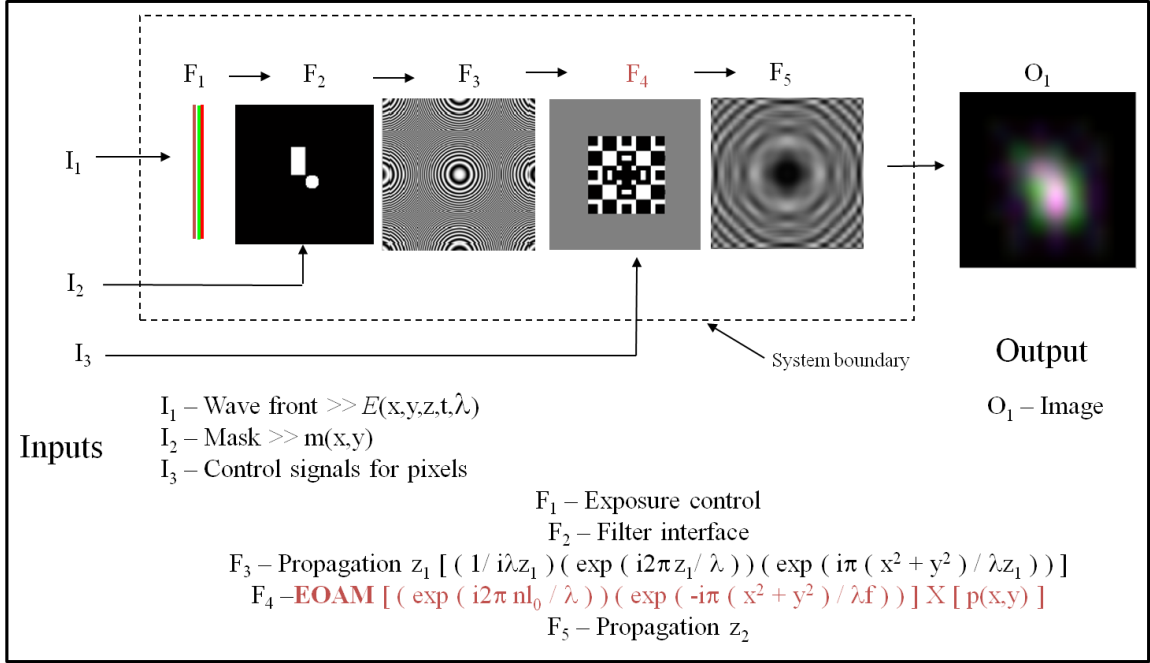


Figure 8: Fresnel Wave Propagation System for Electro-optic Adaptive Microlens

$$I_2 - \text{Mask} \gg m(x, y, z, \lambda)$$

The mask is a physical object that can modify the electro-magnetic wave. The mask is described by $m(x, y, z, \lambda)$ which is a two-dimensional array of complex numbers.

$$I_3 - \text{Control signals for pixels}$$

The EOAM device is controlled by addressing the array of pixels with various applied voltages. The applied voltage determines the relative phase shift introduced into the wavefront by each of the pixels.

$$F_1 - \text{Exposure control}$$

The exposure control function allows for the input wavefront into the system and to transfer energy for a finite amount of time. The function is modeled by a rectangle function, $rect(t)$. The output is the result of $rect(t)$ multiplied by $E(x, y, z, t, \lambda)$.

$$F_2 - \text{Filter interface}$$

The wavefront and mask are input into the filter interface. The interface controls the alignment of the two components and can be modeled by a multiplication of $E(x, y, z, t, \lambda)$ by $m(x, y, z, \lambda)$. The output of the filter interface is a modulated wavefront.

F₃ – Propagation $p(x, y) z_1$

The propagation function models the transfer of the wavefront through a medium. The medium is described by its optical properties, namely optical path distance z_1 , and the propagation is modeled by a convolution with the modulated wavefront.

$$*h(x, y, \lambda; z_1) \quad (3.1.27)$$

$$h(x, y, \lambda; z_1) = \frac{\exp\left(\frac{i2\pi z_1}{\lambda}\right)}{i\lambda z_1} \exp\left(i \frac{\pi}{\lambda z_1} (x^2 + y^2)\right) \quad (3.1.28)$$

F₄ – EOAM

The EOAM element changes the wavefront by modifying the relative phase at each pixel. The model for the EOAM function is the multiplication of the wavefront with the pupil and the EOAM element. The pupil is a two-dimensional complex array that limits the energy transferred from the incoming wavefront to the output wavefront. The EOAM element changes the characteristics of the wavefront.

$$\times \left[e^{\left(\frac{i2\pi n_b}{\lambda}\right)} e^{\left(\frac{-i\pi(x^2+y^2)}{\lambda f}\right)} p(x, y) \right] \quad (3.1.29)$$

F₅ – Propagation $p(x, y) z_2$

The propagation function models the transfer of the wavefront through a medium. The medium is described by its optical properties, namely optical path distance z_2 , and the propagation is modeled by a convolution with the imaged wavefront.

$$*h(x, y, \lambda; z_2) \quad (3.1.30)$$

$$h(x, y, \lambda; z_2) = \frac{\exp\left(\frac{i2\pi z_2}{\lambda}\right)}{i\lambda z_2} \exp\left(i\frac{\pi}{\lambda z_2}(x^2 + y^2)\right) \quad (3.1.31)$$

O₁ – Image

The image is an electro-magnetic field that is known by its electric component, which is a complex vector field, $E(x, y, z, t, \lambda)$. Here the intensity is proportional to electric field squared,

$$I \approx |\vec{E}_0|^2. \quad (3.1.32)$$

2.2 Assumptions for using this propagation model for the EOAM

The model was implemented in Matlab® code with the following assumptions:

- a) Fresnel propagation
- b) Polarization independence
- c) Paraxial region for object and image
- d) Wavelength dependence
- e) Spatially incoherent illumination.

The first three assumptions are based on the theory in section 1.0, Fresnel Propagation.

This code is shown in Appendix A: setupworkspace160_3.m, lensf500bit_3.m, arrayfillbit_160.m, fPropfocal_160.m, and address_fbit.m.

The wavelength dependence of the equations is handled by simulation at three separate values for lambda. The electric field at each wavelength is propagated

completely through the system and the intensity is found at the image plane. The final image is a summation of the three wavelength intensities.

For spatially incoherent illumination each point of the mask can be treated as an independent point source with random phase. In the paraxial region of the system the image can be approximated by the convolution of the point spread function of the system with the appropriately scaled object. The scaling is based on geometric optics for a simple thin lens. Figure 9 and Figure 10 show sample input and output for a focal length 500 micron lens with 256 by 256 micron square pupil having pixel size of 4 microns and dispersion index similar to quartz. The phase has been quantized to 4 levels between zero and 2π , with the regions between pixels approximated by averaging the phase of the adjacent pixels. The object and image distances are based on geometric paraxial Gaussian optics with magnification of -1.25.

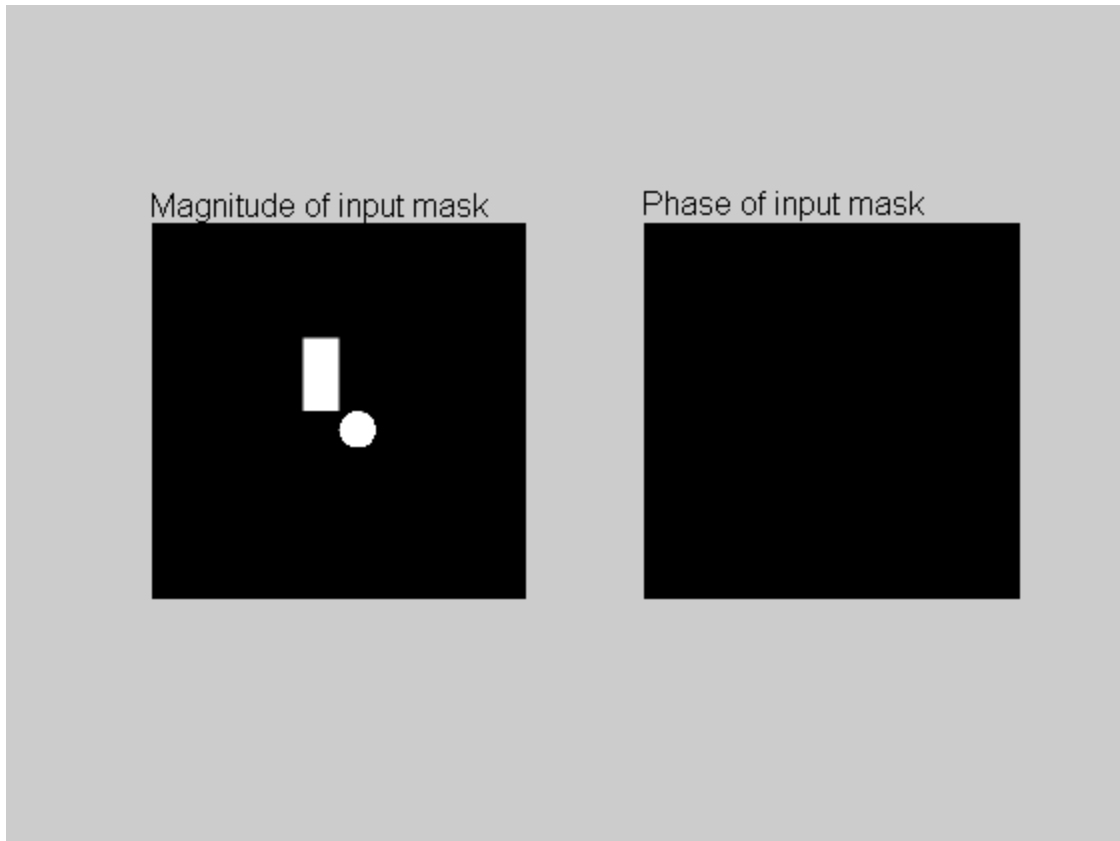


Figure 9: Input mask for EOAM system. Mask represents group of incoherent point sources with random phase and transmission of 1.0 for all wavelengths.

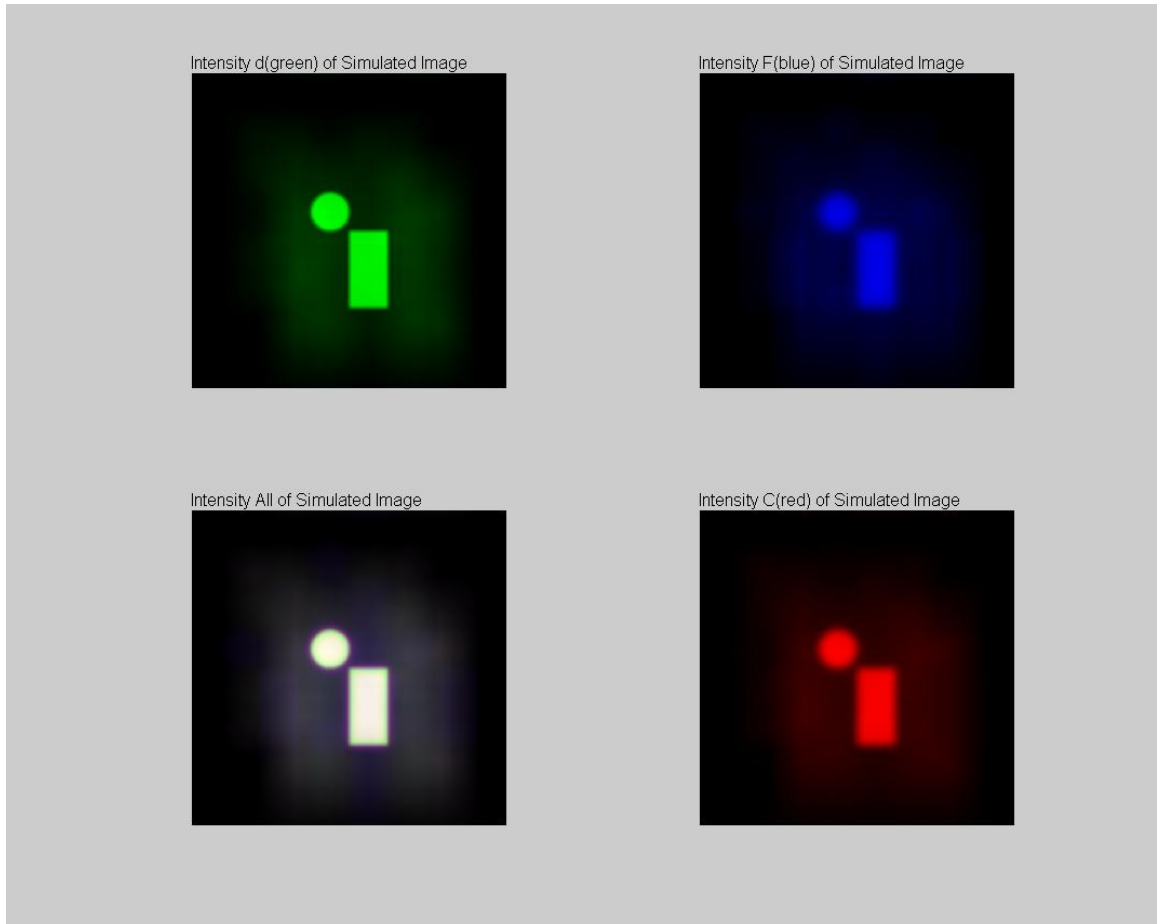


Figure 10: Sample output image for EOAM system. Simulated wavelengths are 587, 486, and 656 nm.

B. Electro-optic Adaptive Microlens materials in single cell device

The EOAM materials properties are determined by mechanical, chemical, electrical, and optical factors. The mechanical and chemical characteristics relate mainly to the fabrication steps involved in building the EOAM device. Because the device was patterned using lithography, the optical properties are also determined during fabrication. The electrical and optical properties are the major contributors to the performance and ultimate utility of the EOAM device in imaging.

The electro-optic material used in the device is a polymer-dispersed liquid crystal (PDLC) described by Ren et al[15]. A mixture of 26% by weight E48 LC and 74% UV curable prepolymer NOA81 was sandwiched between ITO coated glass slides and patterned to create a GRIN PDLC Fresnel lens. Figures 11 and 12 illustrate the structures of the molecular entities of E48 and in NOA81. The mixture was patterned by exposure with UV radiation that induced the monomers in NOA81 to polymerize. The rate of polymerization determines the droplet size of the micro domains of LC material that phase separate as the NOA81 polymerizes. It has been confirmed in the literature [23, 24] that the mean size of the droplets is dependent on the weight fraction of LC and the rate of polymerization.

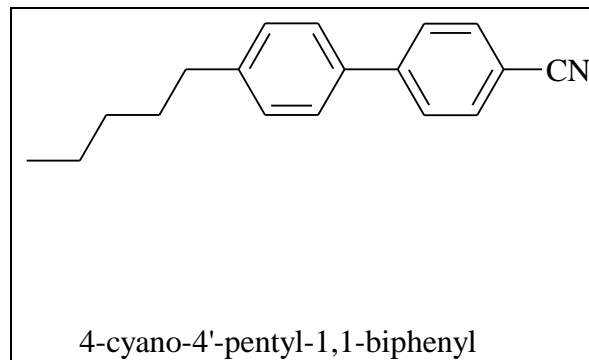


Figure 11: Chemical structure for E48 liquid crystal.

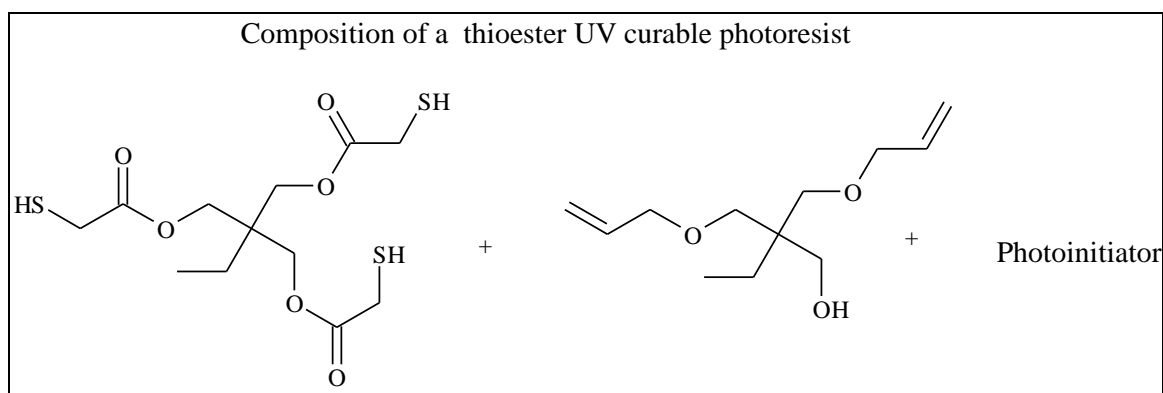


Figure 12: Chemical structure for NOA81.

Several composite films (Samples 001-005) with 10 micrometer polystyrene spheres used as spacers on the substrate were cured and used for preliminary studies on single cell devices. The samples were exposed with a mercury arc source filtered at 365 nm. Because the droplet size of the micro domains of LC were slightly larger than the wavelength of illumination, Sample 001, exposed at intensity of $\sim 2.25 \text{ mW/cm}^2$, appeared somewhat milky in color under white light illumination and the LC droplets scatter the light. In Sample 005, exposed at intensity of $>50 \text{ mW/cm}^2$, the droplet size was equal to or smaller than the wavelength of visible light and appeared clear. Figure 13 shows transmission data collected on the two samples. Sample 005 had higher transmission but exhibits some loss of transmission at shorter wavelengths. This indicated that the droplet size was approximately on the order of the blue wavelengths.

The next generation of single cell devices (shown in Figure 14) were fabricated utilizing patterned SU-8 photoresist as a spacer on the ITO glass slides and the E48/NOA81 precursor exposed at intensity of $>50 \text{ mW/cm}^2$.

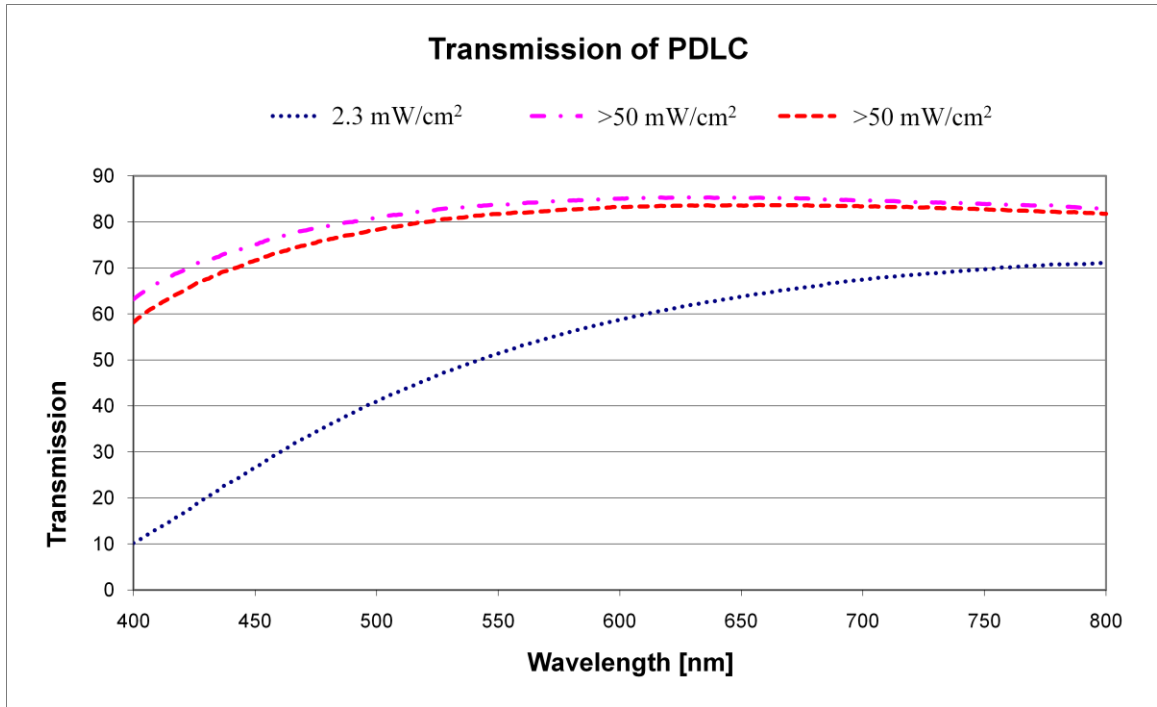


Figure 13: PDLC at 24% by weight in NOA81 polymerized at different intensities. Sample transmission data includes losses due to two ITO coated glass slides.

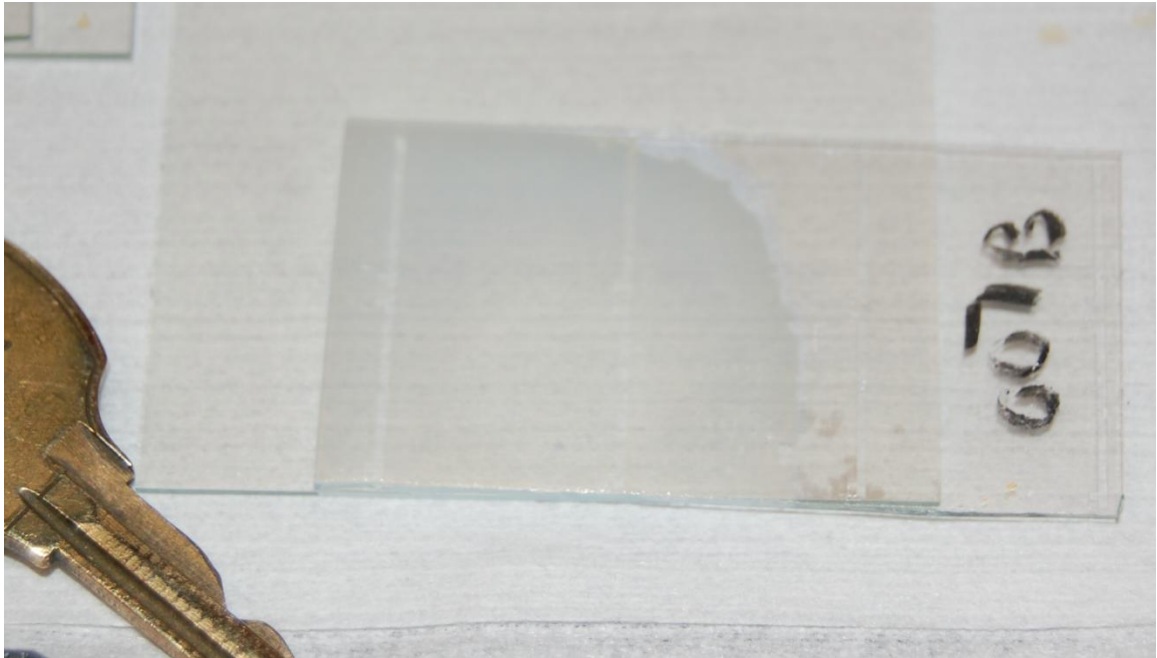


Figure 14: Image of single cell device made from E48/NOA81 utilizing patterned SU-8 resist as the spacer.

The samples were also evaluated using a Michelson interferometer to determine the relative phase shift induced by an applied field. The samples were inserted into the

measurement arm of the Michelson interferometer and a voltage was applied to the ITO layer of one slide, while the other ITO slide was grounded. The light path made a double pass through the sample with this configuration as shown in Figure 15.

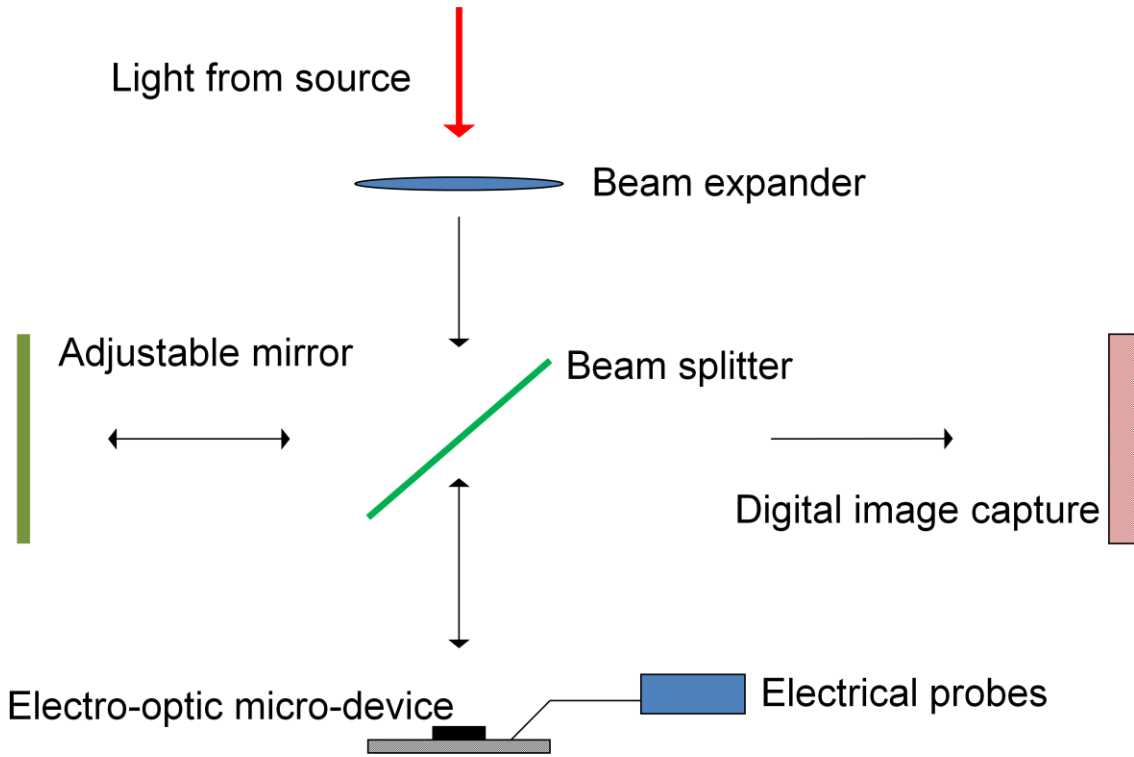


Figure 15: Diagram of dual beam interferometer test system used for single cell devices.

The interferograms from the dual beam interferometer were captured on a digital camera back and analyzed using code written in MATLAB®. This code (Appendix B, `datain_rerun_singlefile.m`) allowed selection of an area of multiple interferograms for analysis of minimums and calculation of the relative shifts of the minimums. The interferograms were captured with different applied voltages and a sample of the output phases fit to 3rd order splines is shown in Figure 16. A system to measure phase for reflective electro-optical micro-devices at visible wavelengths was presented at Optical Fabrication and Test Conference in 2006 [25].

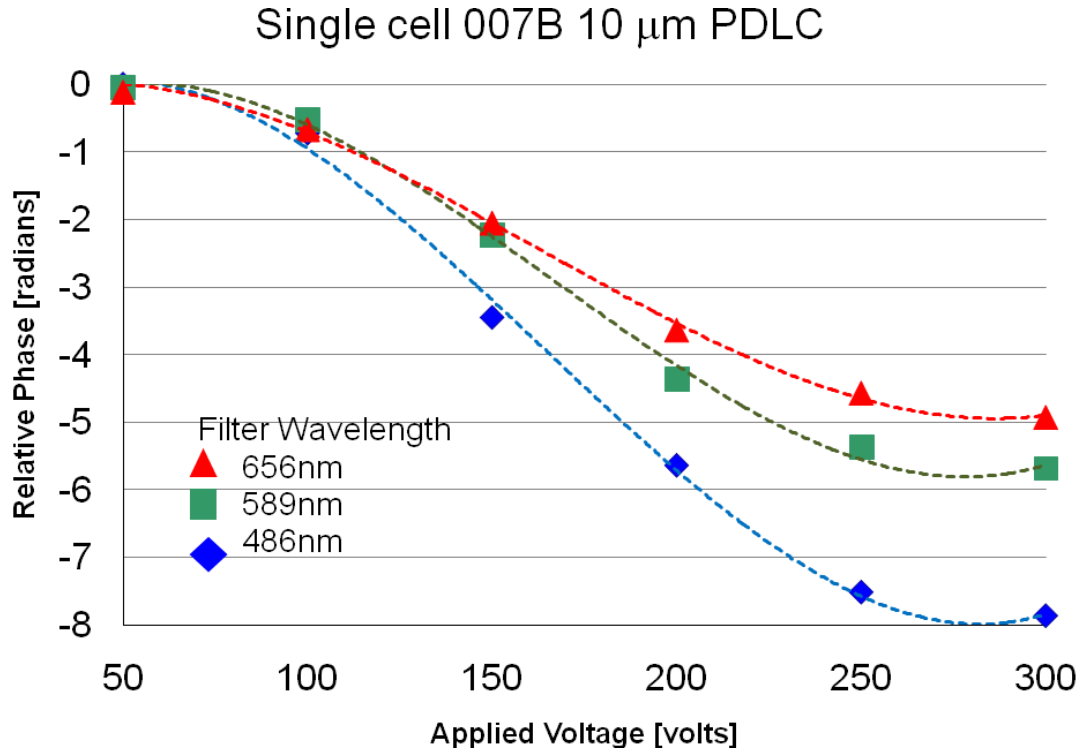


Figure 16: Phase shift versus voltage for single cell device at three wavelengths.

The single cell device was also evaluated at the red wavelength for sensitivity to polarization of the illumination. The PDLC as fabricated is reported to be polarization independent [15]. Interferograms were captured for the device with a linear polarizer inserted in the beam path ahead of the beam splitter. The resulting relative phase versus applied voltage is shown in Figure 17. A 100%(1- α) confidence interval, with $\alpha = 0.05$, was constructed on the regression of Relative Phase on Applied Voltage for P270. This confidence interval was used to compare the 3rd order fit for P0 to the 3rd order fit for the P270 regression. It was found that the estimated fit for P0 fell entirely within the 95% confidence interval for P270, so no significant difference between the two processes can be discerned at the $\alpha = 0.05$ level. Note that this confidence interval is a realization of all possible intervals at this probability; this indicates that 95% of the time the true

regression (not the individual observations) will fall within the estimated upper and lower bounds of the interval.

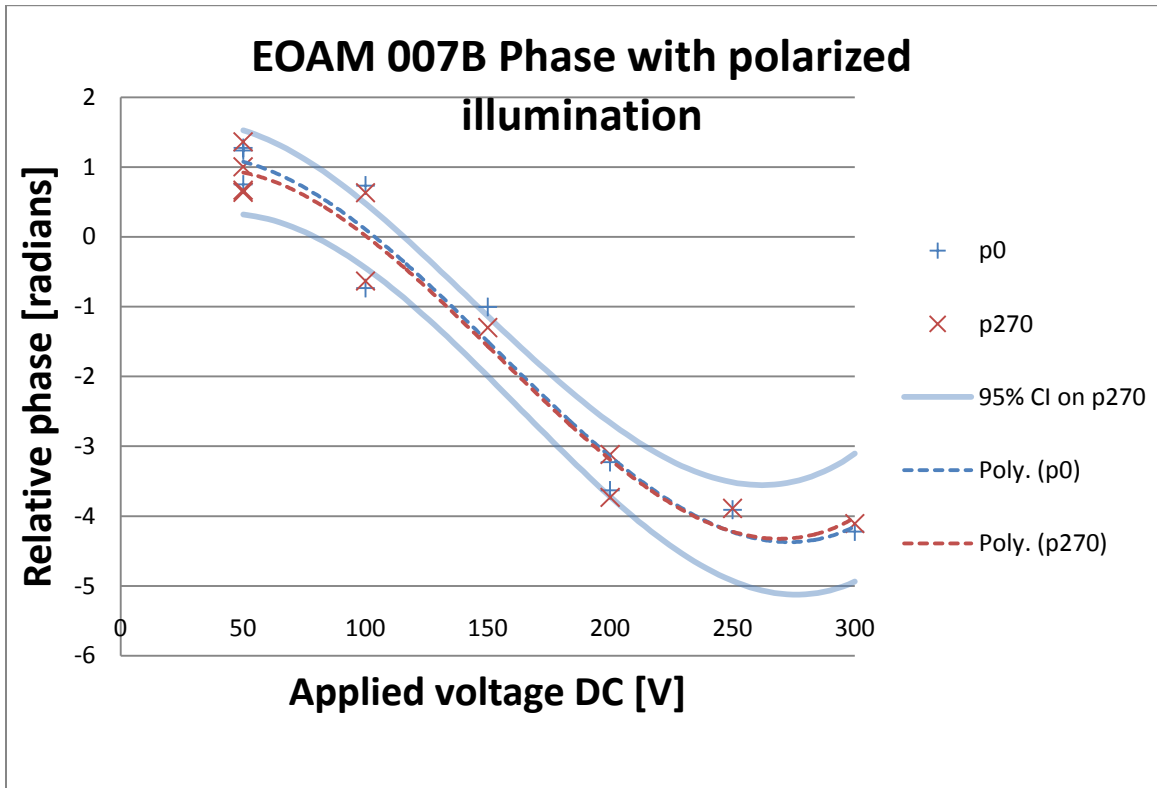


Figure 17. Comparison of phase shift with illumination of two different polarizations.

C. Control voltages for the arrayed pixel device

The applied field for each pixel of the EOAM controlled the relative phase for that pixel. The simulation program (Appendix A, address_fbit.m) for addressing of pixels in the device array at various phase levels was written to incorporate the relative phase shift as a function of the material thickness, the applied field and the wavelength of illumination. The number of required phase levels for operation of the system was quantized and also incorporated in the simulation.

To simplify the addressing of the device the work utilized only two phase levels. The imaging performance of the EOAM system can be enhanced by allowing more phase

levels. A voltage control device that allowed quantization of the applied field at 2^2 levels was built on a breadboard for use in testing the EOAM devices.

The power distributor design was based on the need to use one power supply with multiple outputs. The EOAM devices use applied voltage over a range of 0 to 300 volts AC or DC but do not conduct current. Accordingly, the distributor was designed for safety to limit current to less than 1 mA. If the EOAM device has current flow, it has failed and is no longer useable. Slow blow 10 mA fuses were assembled in-line on the power source leads, and ~300 k ohms resistors were added to the EOAM leads. The power supply varied from 0 to ~240 volts, and the distributor board was designed to have four levels plus ground. Figure 18 shows a diagram of the voltage distribution board. The voltage at each node for the EOAM is as follows:

$$V_{out} = V_{in} \left[R_1 / R_1 + R_2 \right] \quad (3.2.1)$$

where R_1 is the resistance before the node and R_2 is the resistance after the node with the input voltage of V_{in} .

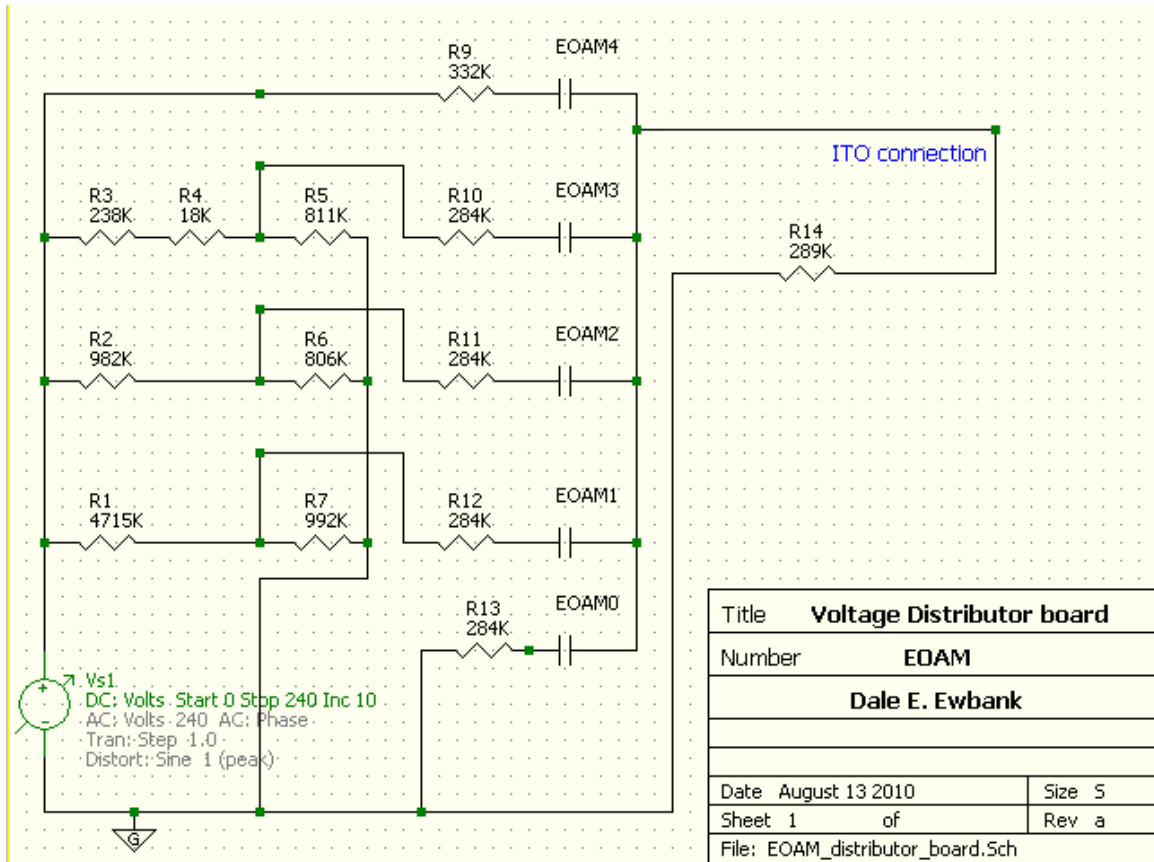


Figure 18: Diagram of voltage distribution board for EAOM device.

DC voltage was used in testing the single cell devices and the first fabrication run of the 16 by 16 pixel devices. As the single cell and pixel devices had a long memory (> 30 seconds to return to off state) with DC applied voltage, the subsequent pixel devices were tested using an AC source at 60 Hz.

D. Fabrication process for the arrayed pixel device

The fabrication processes for the EOAM made use of both newly developed and existing processes within the Semiconductor and Microsystems Fabrication Laboratory at RIT. Detailed instructions for the fabrication processes are given in Appendix C, Run Sheet for EOAM v1.5b, and illustrations of the fabrication processing steps are shown in Appendix D, EOAM_Process Rev1_5b.PPT.

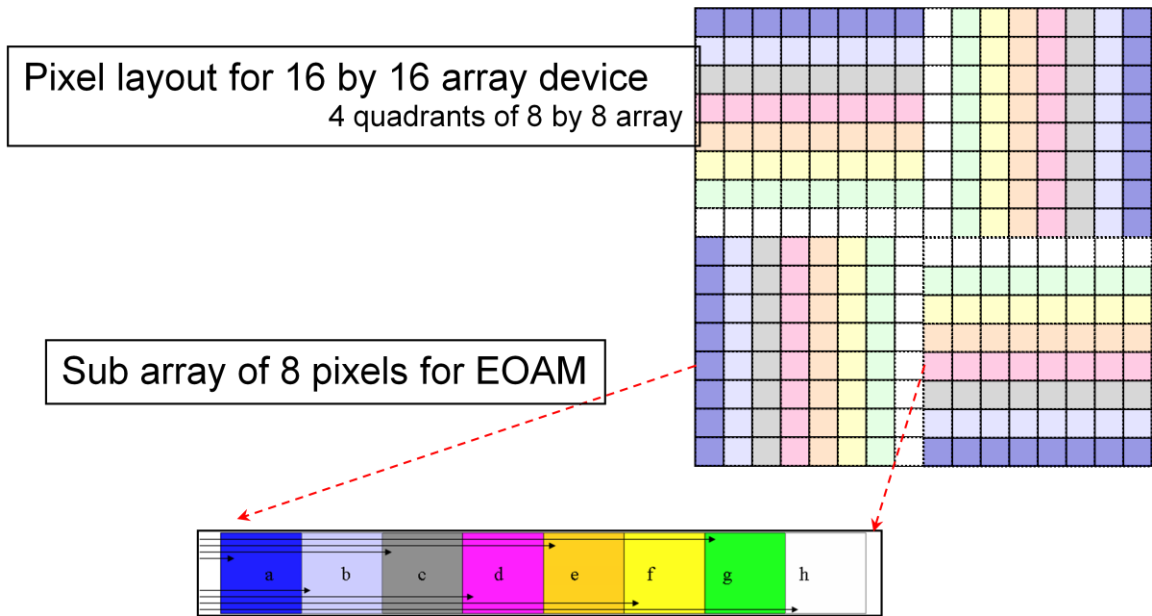


Figure 19: Sub array of pixels for EOAM device with 16 by 16 pixels.

Figure 19 shows a sub array of 8 pixels (a through h) that are addressed individually. These are built up to create the four quadrants of the device. The routing layout for the device was optimized for the sub array. For a lens that is circularly symmetric to the optical axis, this arrangement of cells allows for multiplexing of the cells that are of equal radial distance from the optical axis (usually the center of the device). The code 'address_fbit.m' calculates the quantization levels for the lens phase and thus the addressing needed for each pixel. Figure 20 shows an example of the

addressing a single quadrant for a 250 mm focal length microlens for phase quantized at 2 and 4 levels.

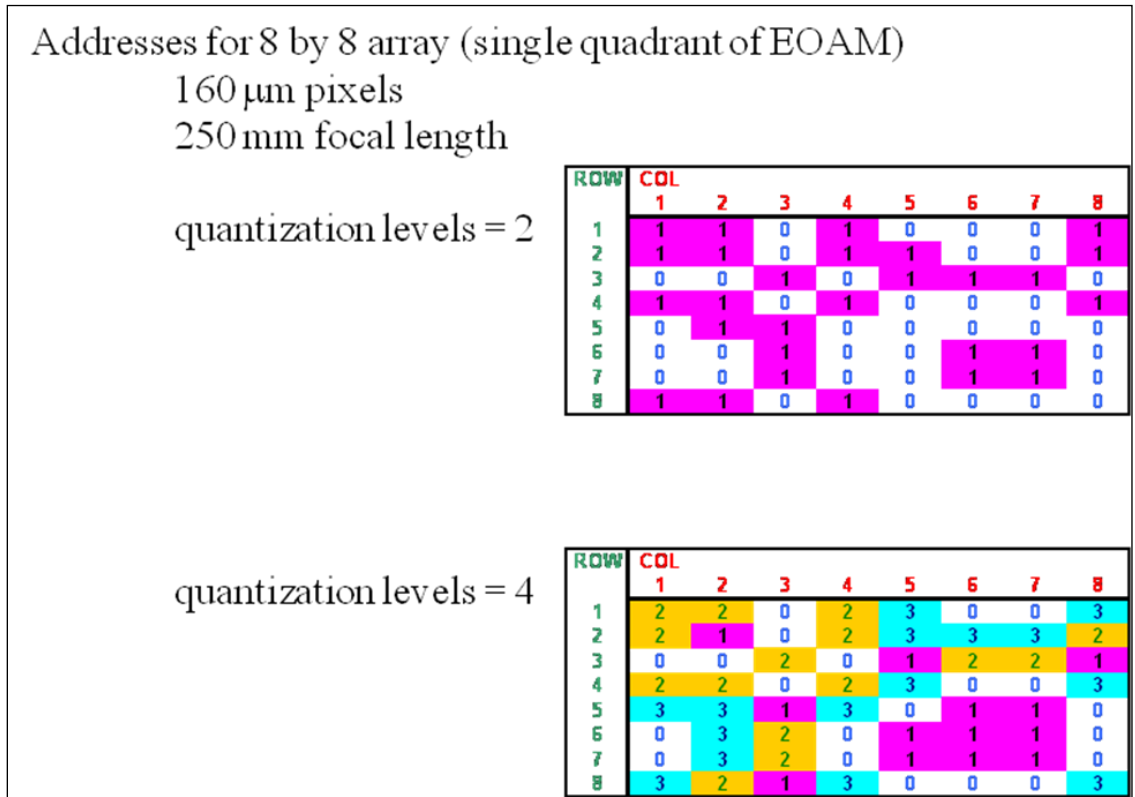


Figure 20: Addresses for 8 by 8 array of a single quadrant with 160 micrometer pixels.

The design for wiring to bond pads utilized ICgraph by Mentor Graphics Corporation. Four sets of bonding pads are on the device, however access is only needed to one set. The others are redundant and also allow for redundant multiple contacts between the metal layers.

Figures 21 through 24 show the layout of the EOAM device levels.

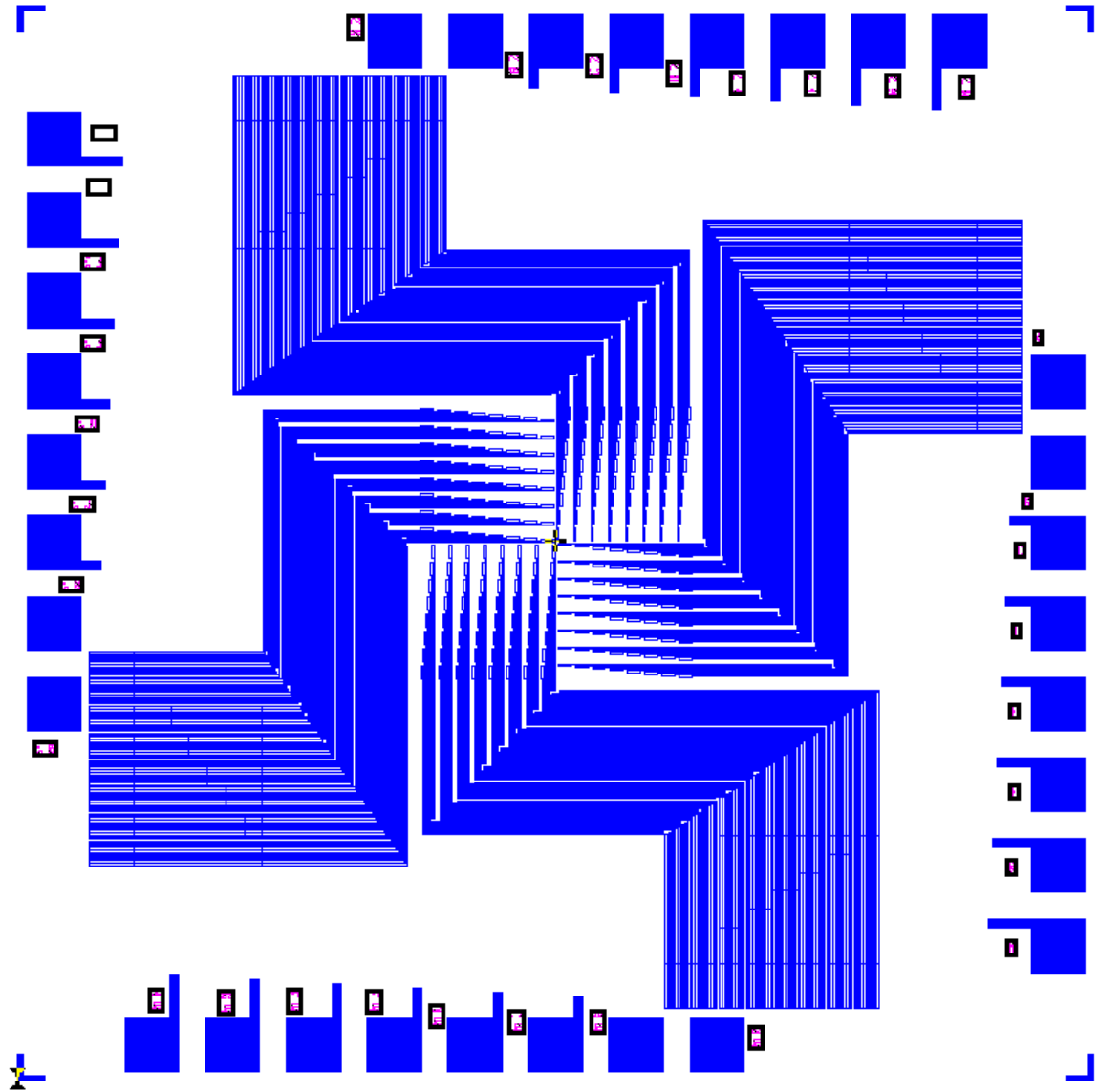


Figure 21 Metal 1of EOAM.

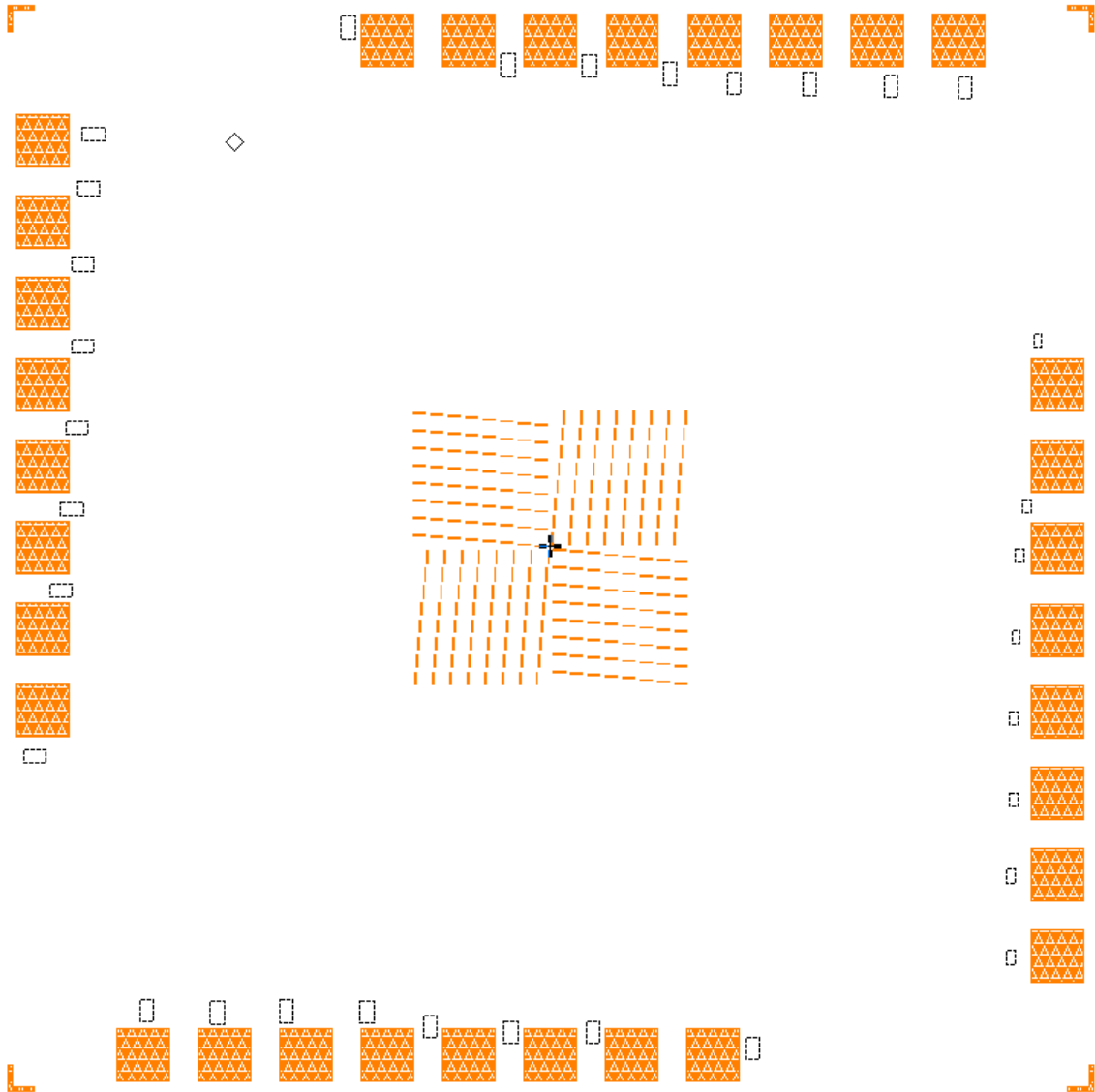


Figure 22: Via 1 of EOAM.

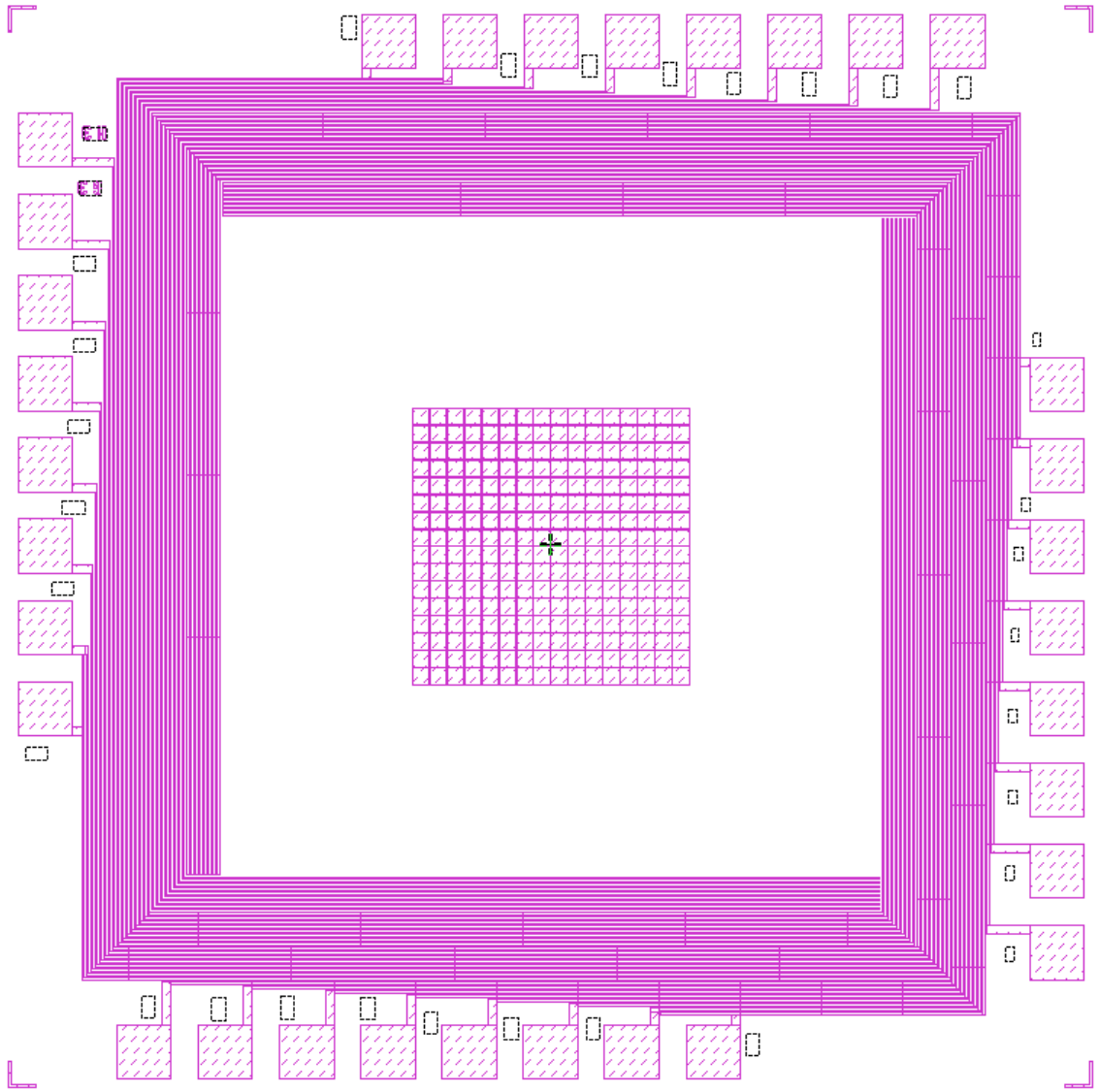


Figure 23 Metal 2 of EOAM.

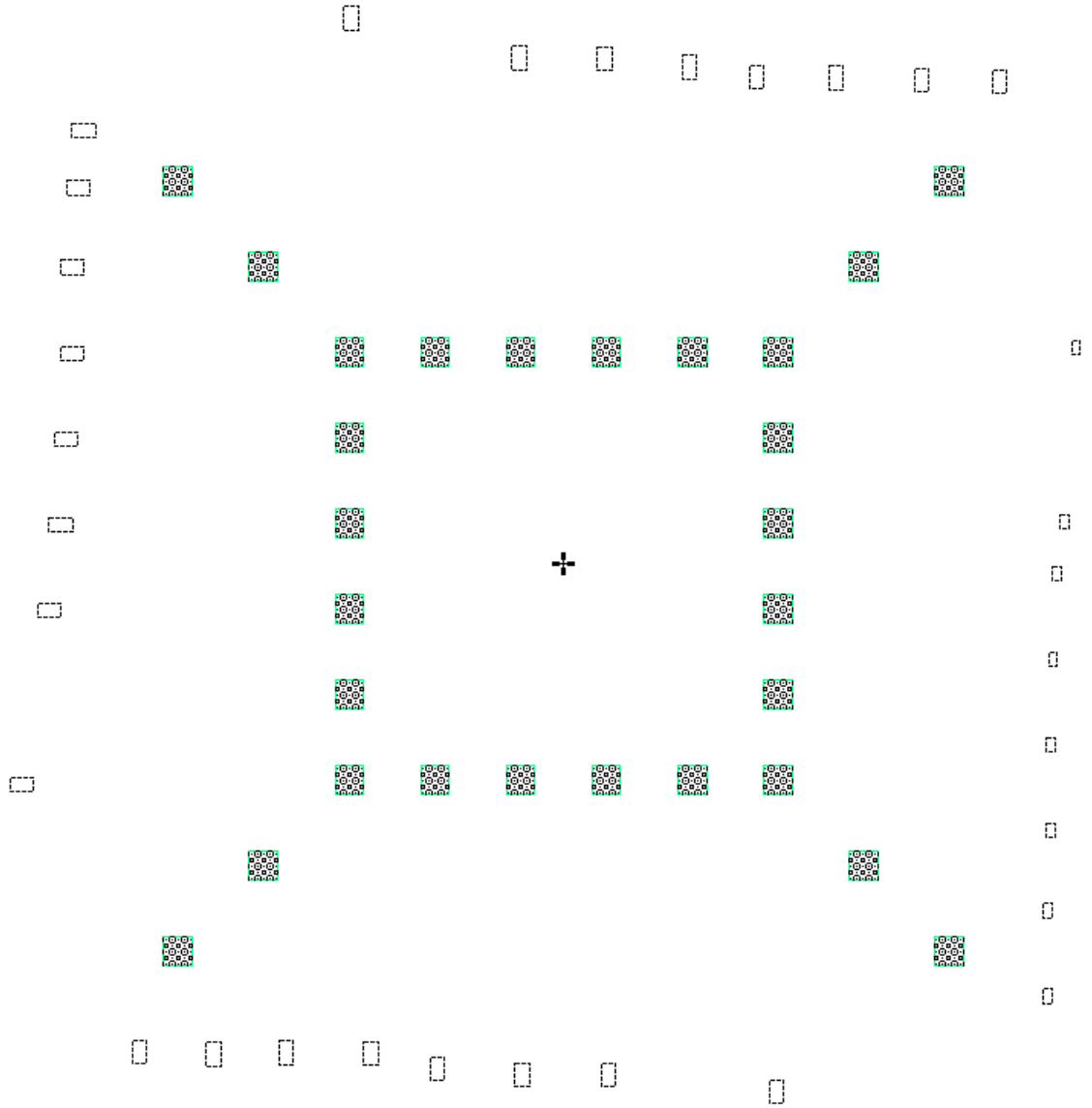


Figure 24 Spacer layer for EOAM.

The fabrication process had 4 levels that required microlithographic patterning. Figure 25 is a diagram of a complete layer stack for one pixel. Metal 1 is the wiring for addressing the pixels. Metal 2 is the conductor that defines the lower electrode for each pixel. The ITO on the glass cover was attached to the ground electrode.

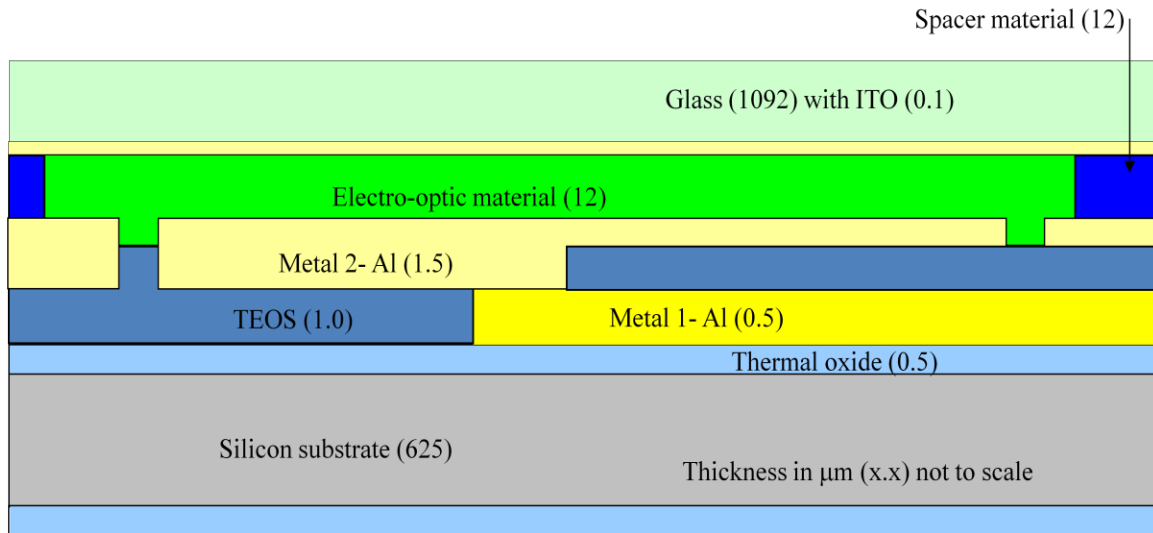


Figure 25 Layer stack for EOAM device.

After fabrication of the first devices it was found that the metal 2 layer had a surface roughness that was too large and the variation in surface height of the pixel due to the contacts resulted in large optical path variations. This layer is the reflective mirror in the device and as such, should have specular rather than diffuse reflections. A new process was developed and implemented to polish the metal 2 surface. The thickness of the deposited aluminum was increased to 1.5 micrometers from 1.0 micrometers to ensure step coverage over the metal 1 and contact edges, and still allow for removal of metal 2 material. The chemical mechanical planarization (CMP) of the metal 2 was done on the Strausbaugh. This CMP process was optimized and was unique in the fact that the aluminum layer was patterned before CMP. If the metal was planarized first, then alignment to the underlying features would have been very difficult.

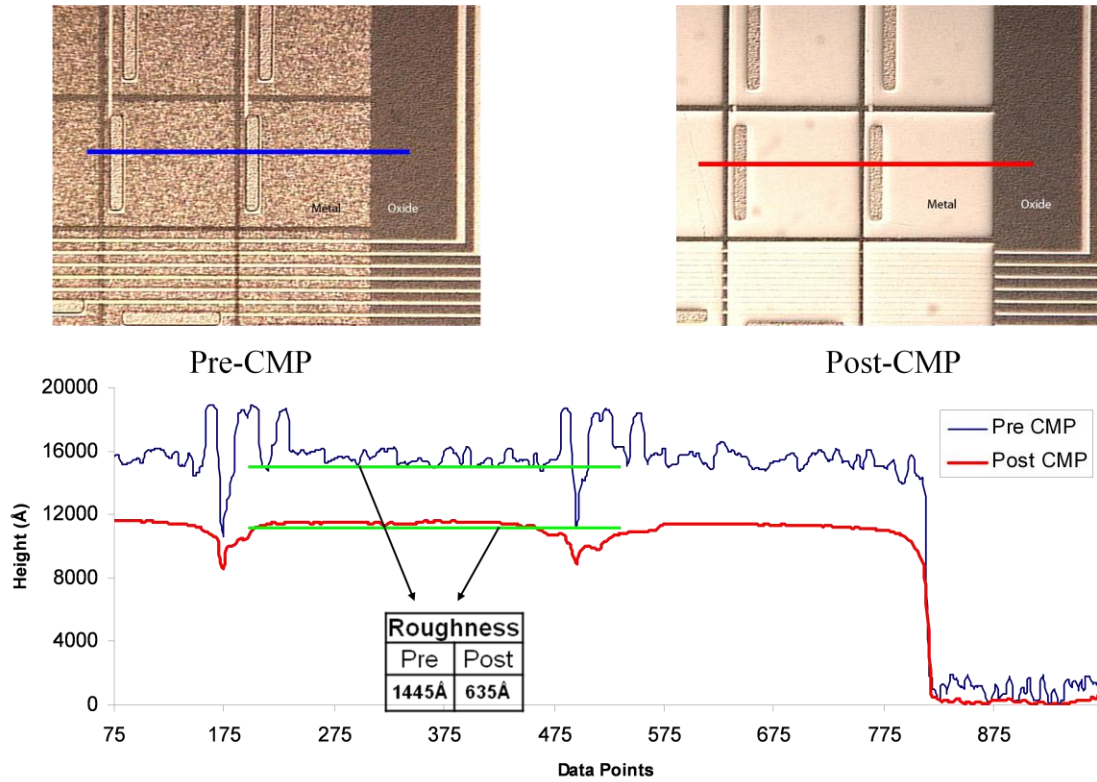


Figure 26: Examples of surface roughness before and after the CMP process [26] on device pixels.

The devices were probed prior to CMP to verify the proper conductivity between the pads and the appropriate pixels for the device design. This allowed for sorting of good devices before CMP and also prevented scratching of the planarized surface.

After CMP the spacer layer was patterned in SU-8 photoresist. The SU-8 is a negative photoresist and was cross linked with exposure at i-line and post development baking. A sample wafer completed to this step is shown in Figure 27. The SU-8 pattern established the region in which the electro-optic material would be deposited.

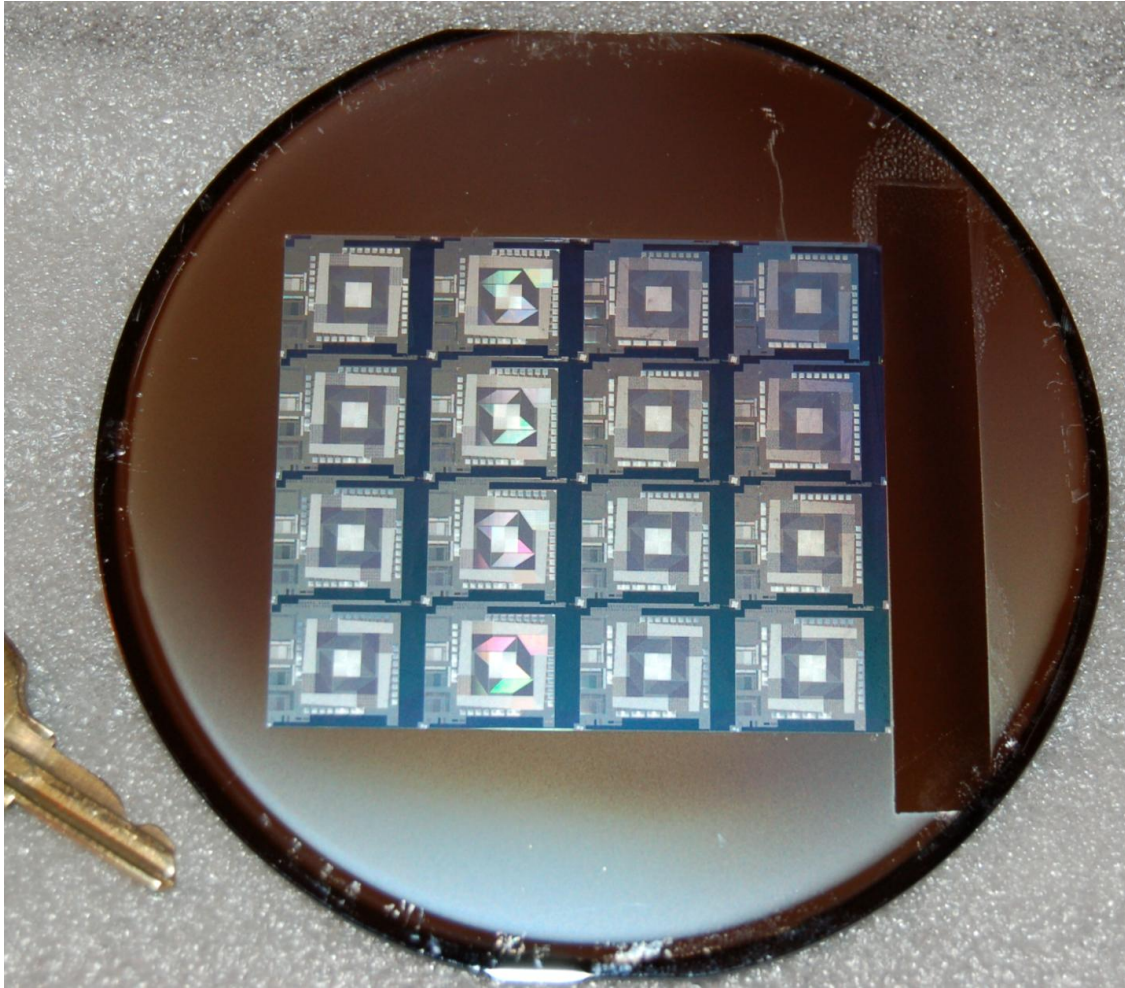


Figure 27: Hundred millimeter device wafer after all lithography steps completed.

The E48/NOA81 mixture was deposited as a liquid and covered with the ITO glass slide. Photopolymerization of the PDLC bonds the ITO glass slide in place. A specialty UV exposure tool was assembled for use in photopolymerization of the E48/NOA81 mixture. This tool allowed for controlled intensity and thus control of the rate process for droplet formation of the LC domains. Figure 28 show a finished EOAM device.

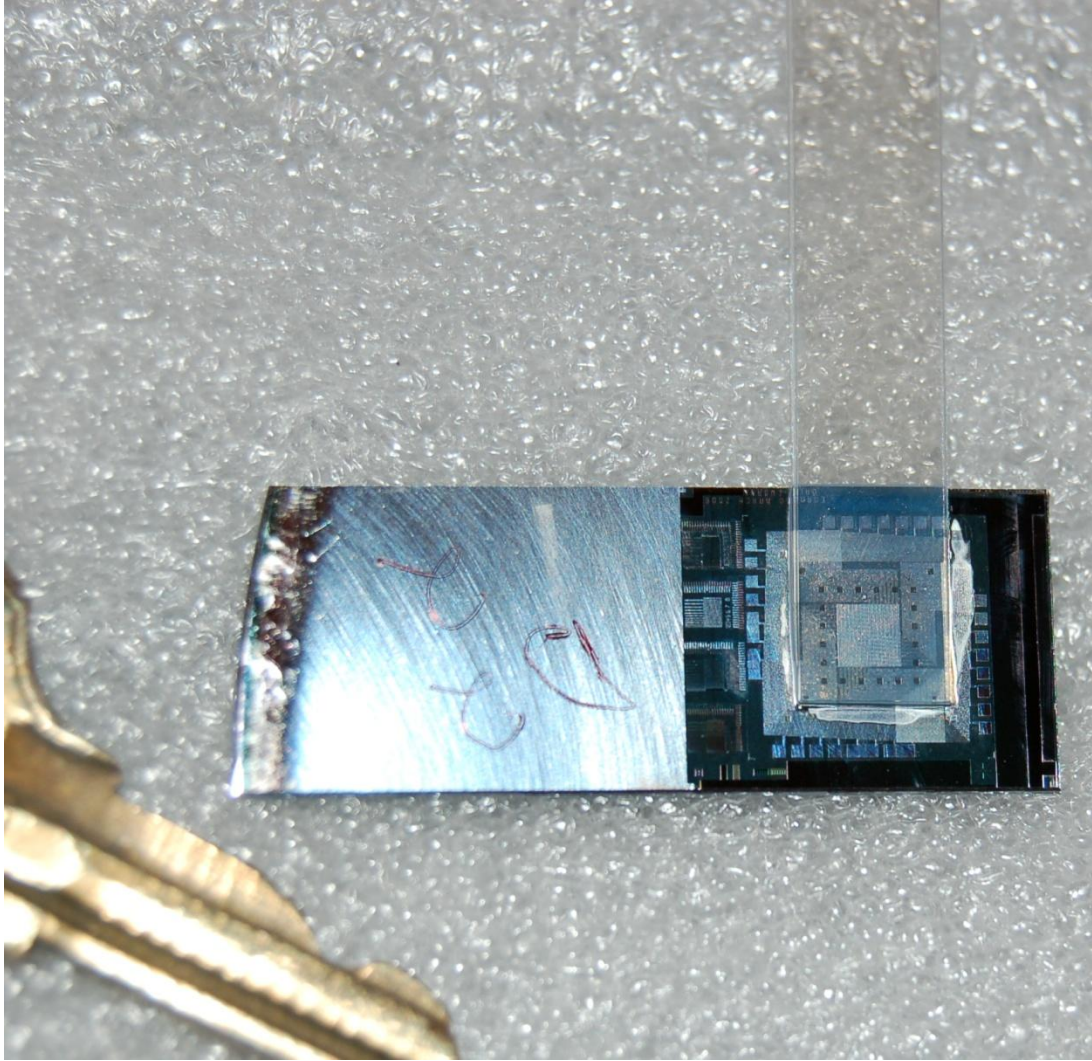


Figure 28: Finished EOAM device.

Detailed instructions for the fabrication processes are given in Appendix C, Run Sheet for EOAM v1.5b and illustrations of the fabrication processing steps are shown in Appendix D, EOAM_Process Rev1_5b.PPT.

Arrayed pixel device builds

The fabrication of the arrayed pixel EOAM devices was carried out with three iterations of device build runs. The first iteration validated the masks and the processing steps that were used. The initial masks for Metal 1 and Metal 2 had minor design errors

and were rebuilt. The first iteration of EOAM devices could only be tested with all pixels addressed to same voltage. The surface roughness for the Metal 2 layer was also identified with the initial run.

The second iteration implemented the new masks, the CMP process, and adjusted film thicknesses of the layers to allow CMP. The third iteration repeated the processes used and greater care was taken to reduce contamination defects and improve yield to result in more devices for testing.

E. Device Testing

1. Dual Beam Interferometer

While each new build was ongoing, devices from the previous run were used for evaluation of phase versus applied voltage. The phase extraction method of using multiple interferograms with various applied voltage for analysis of minimums and their relative shifts as used for the single cell device was implemented for the pixel arrays addressed with same voltage. The results from this method proved to be unreliable. Changes in the optical path lengths of the dual beam interferometer system due to vibration and ambient temperature fluctuations caused the relative phase shifts to be unrepeatable.

Figures 29 and 30 show a device in the dual beam interferometer test system. Figures 31 through Figure 33 are interferograms captured for a pixel device with all pixels address by same voltage. Figure 34 shows a plot of data from analysis of the interferograms from the dual beam interferometer. The line connects the data point in the order that the interferograms were collected (total time of about 10 minutes). As can be seen, the output phase is not predictable.

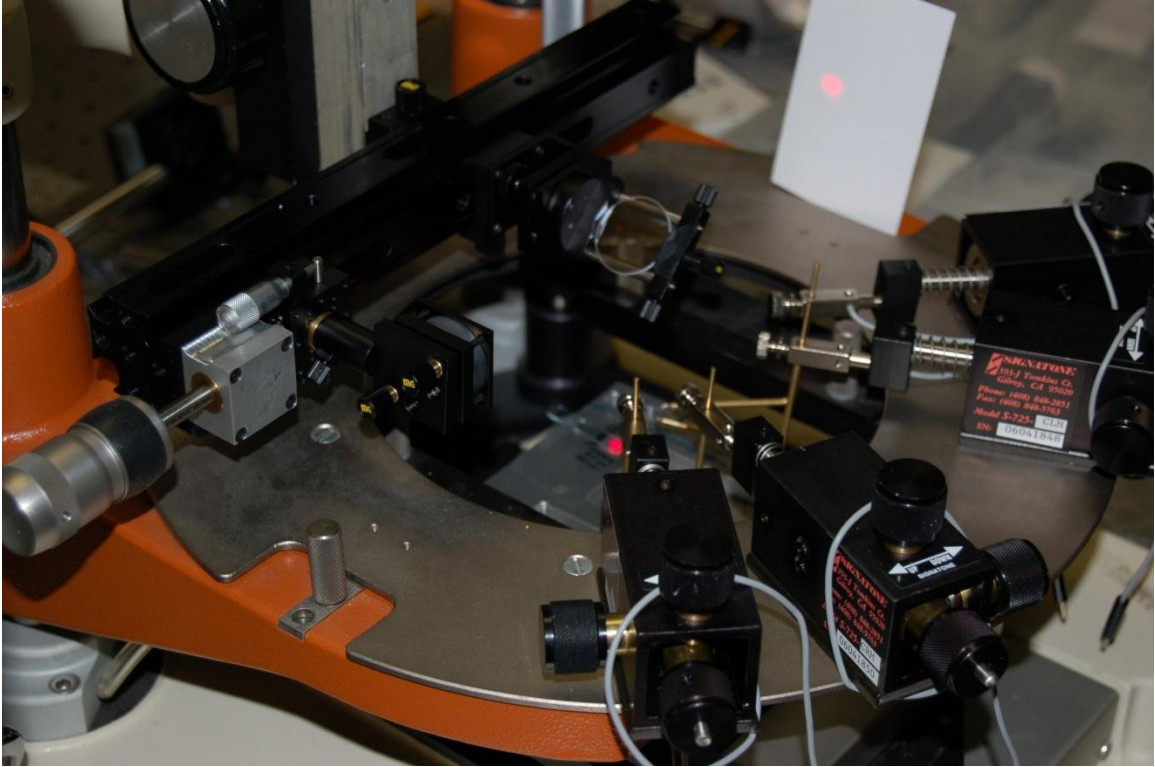


Figure 29: Dual beam interferometer test system with device and output image on card.

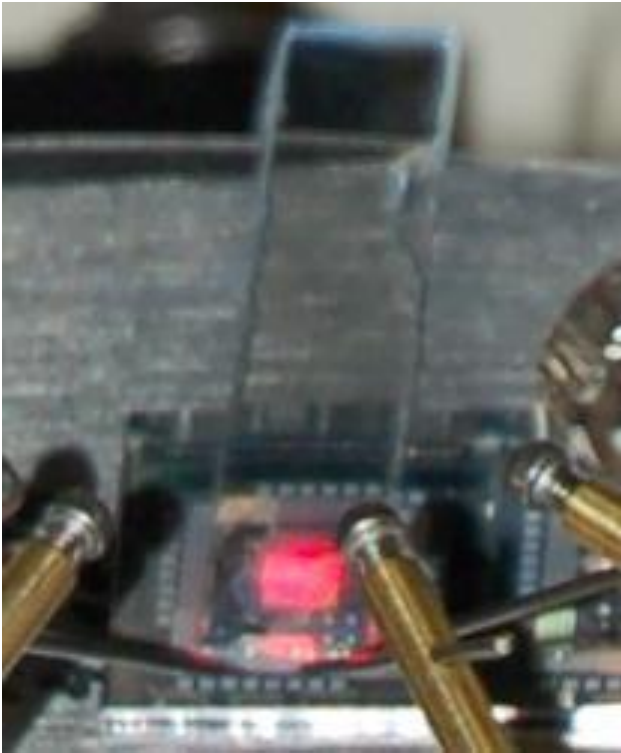


Figure 30: Device with probes in dual beam interferometer test system.

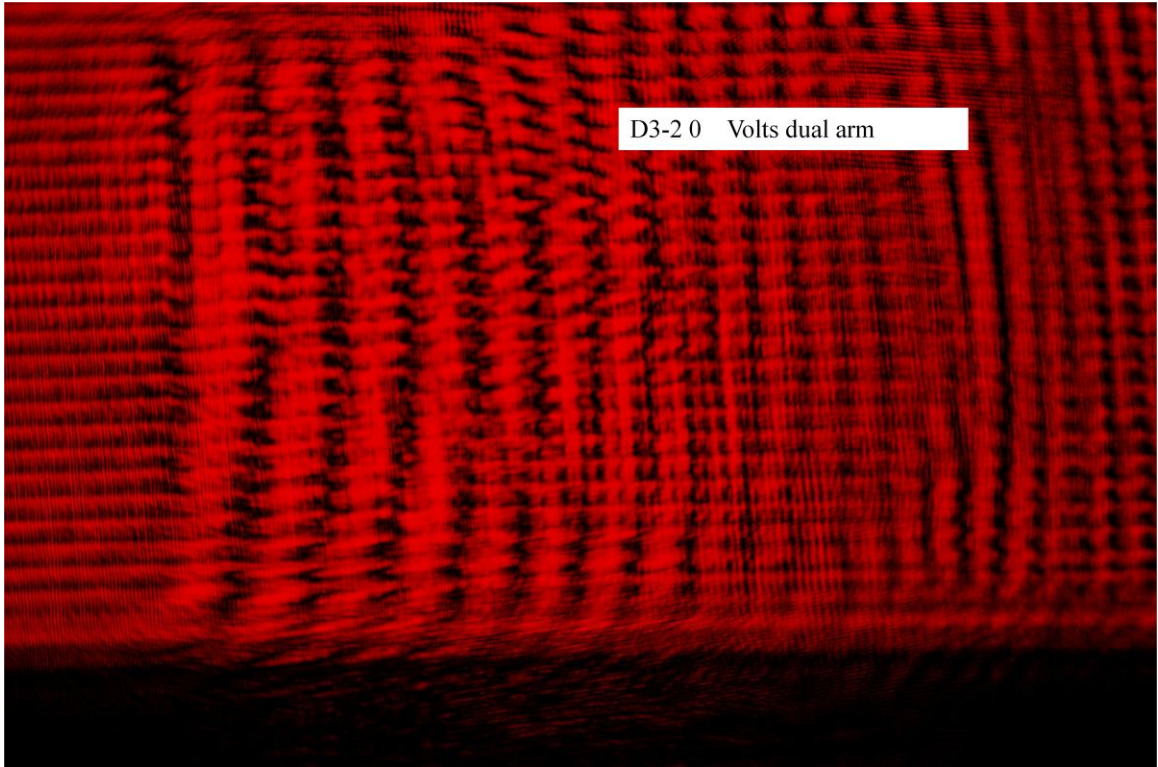


Figure 31: Interferogram of device from dual beam interferometer test system.

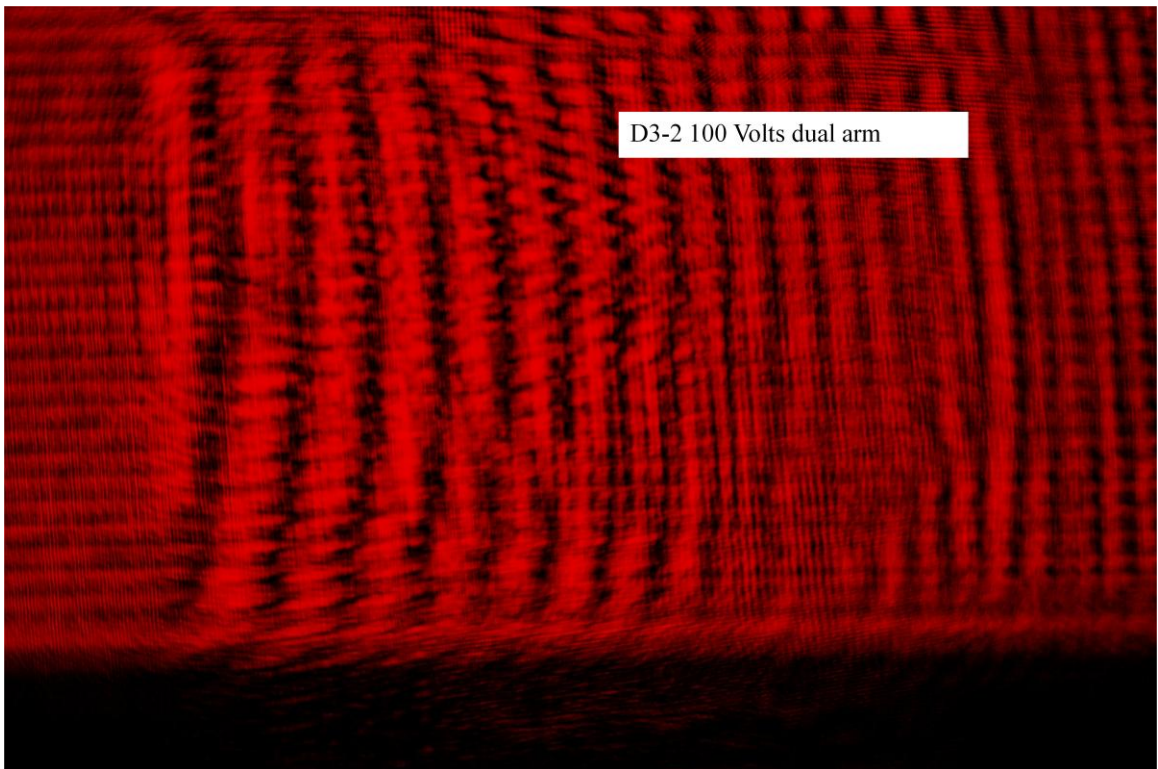


Figure 32: Interferogram of device from dual beam interferometer test system.

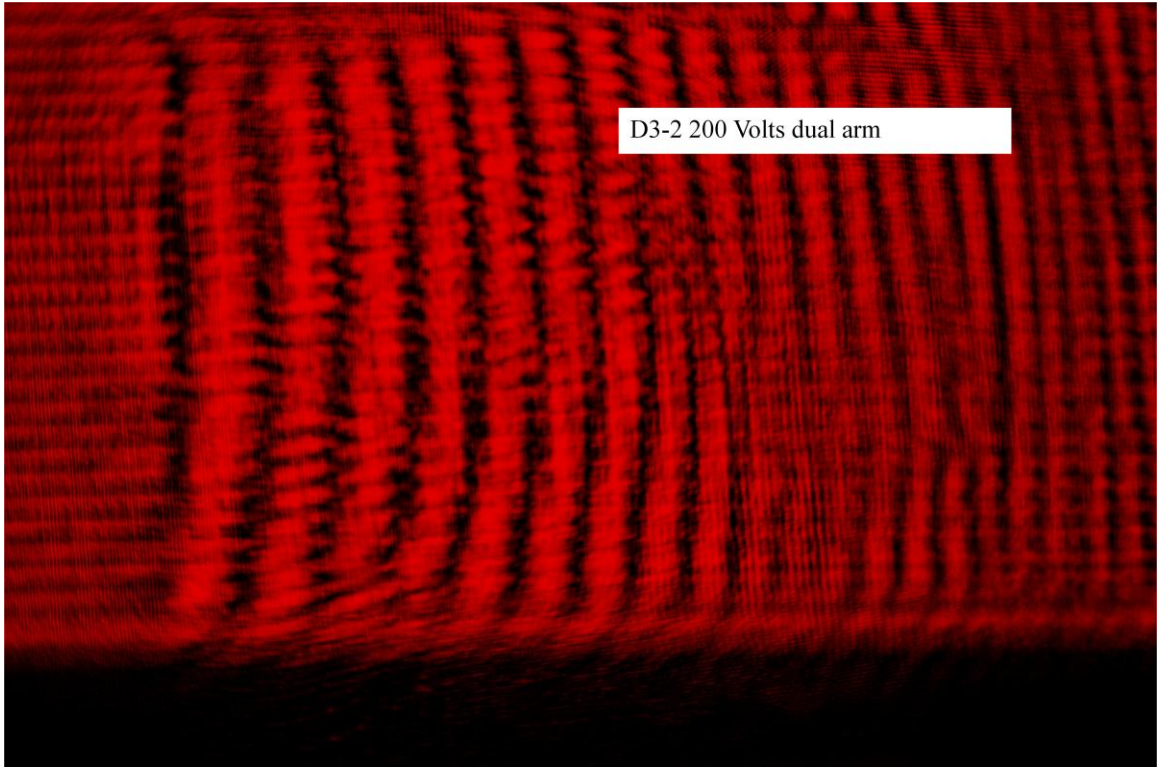


Figure 33: Interferogram of device from dual beam interferometer test system.

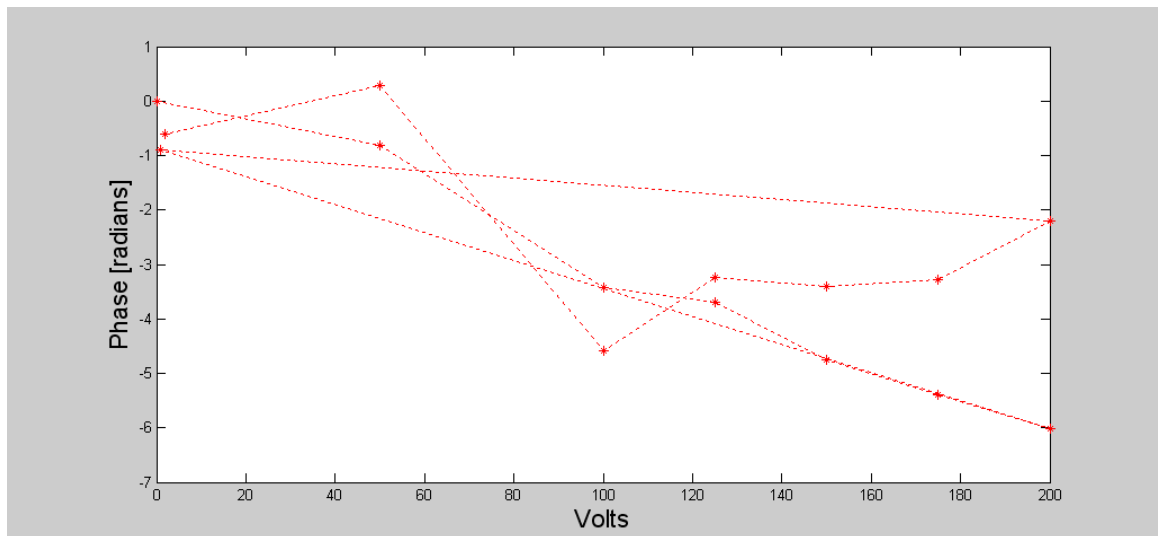


Figure 34: Plot showing relative phase variability from dual beam interferometer test system analysis.

2. Single Beam Interferometer

a. Theory

A new method for extraction of the phase was needed. During the set up of the dual beam interferometer system it was noted that interference patterns could be generated by utilizing only the EOAM device in a single arm of the interferometer. The “surface” reflection of the device was known to exist; but had been considered as a nuisance reflection and needed to be minimized as it contributed to the ‘noise’ component during device measurement and usage. With the dual beam system the reference beam wavefront and the device beam wavefront were to recombine and generate the appropriate interferogram; the “surface” reflection wavefront was 5 to 10 times less intense and to be neglected.

The reference beam mirror was taken out of the dual beam interferometer system to create the single arm interferometer system. The EOAM device acts as a Pohl fringe-producing system [22] that can generate interferograms. The use of a single arm interferometer system for reflective micro-device phase measurement was presented at Optical Fabrication and Test Conference in 2010 [27]. Figure 35 shows a diagram of the single arm interferometer system used.

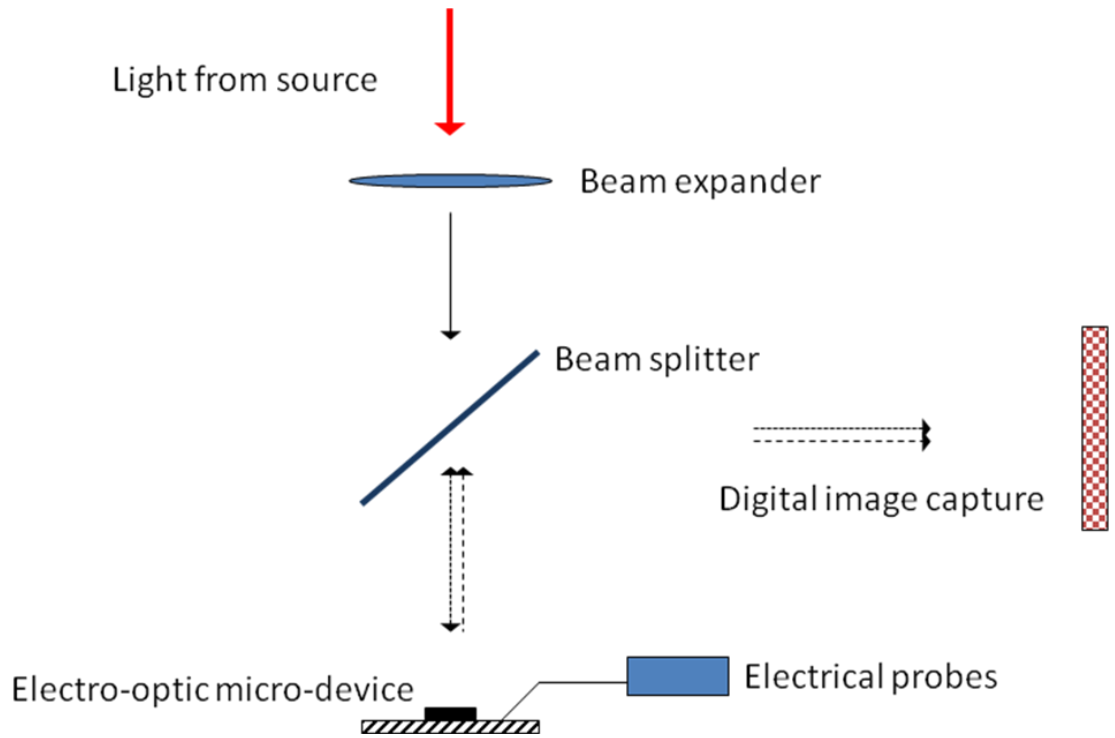


Figure 35: Single arm interferometer system diagram.

Figures 36 and 37 show interferograms from the device tested in the single arm interferometer system. The interferograms include the active device area and the surrounding non-active electrical pad area (right side of image); the non-active area interferogram does not change with applied voltage. The non-active areas were masked with black tape as shown on the EOAM device in Figure 38.

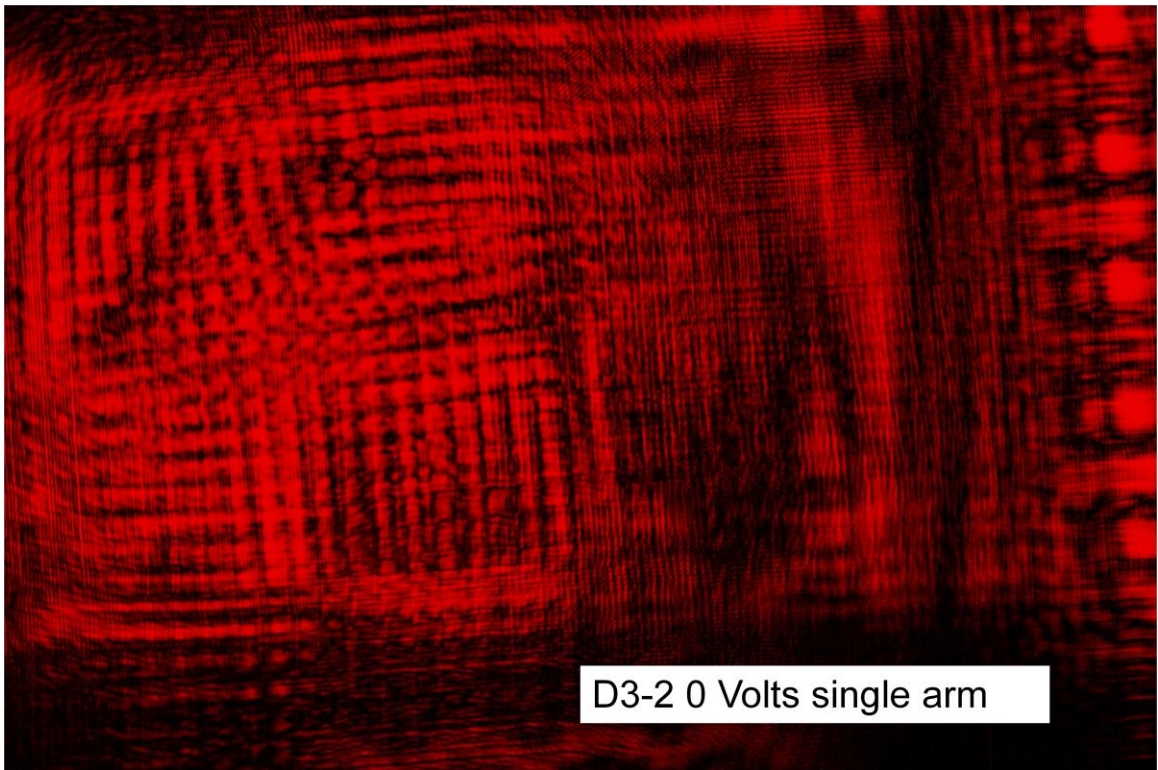


Figure 36 Interferogram from EOAM device active area and surround at 0 volts.

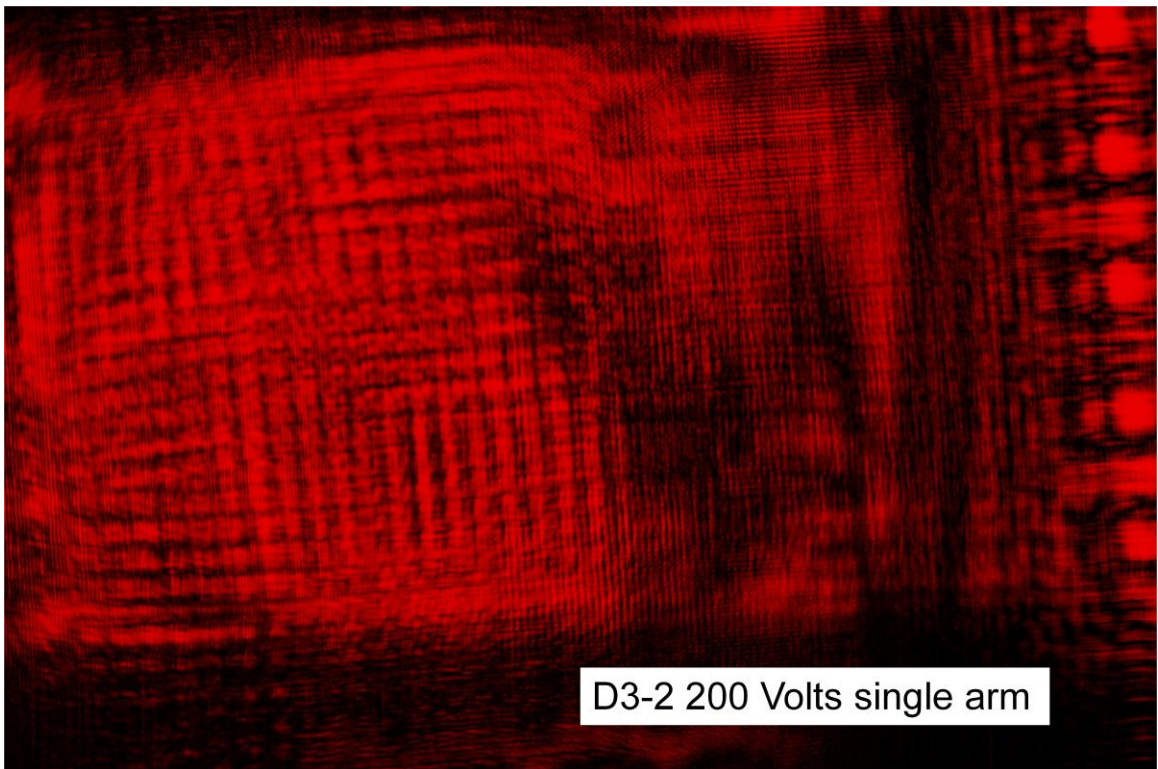


Figure 37 Interferogram from EOAM device active area and surround at 200 volts.

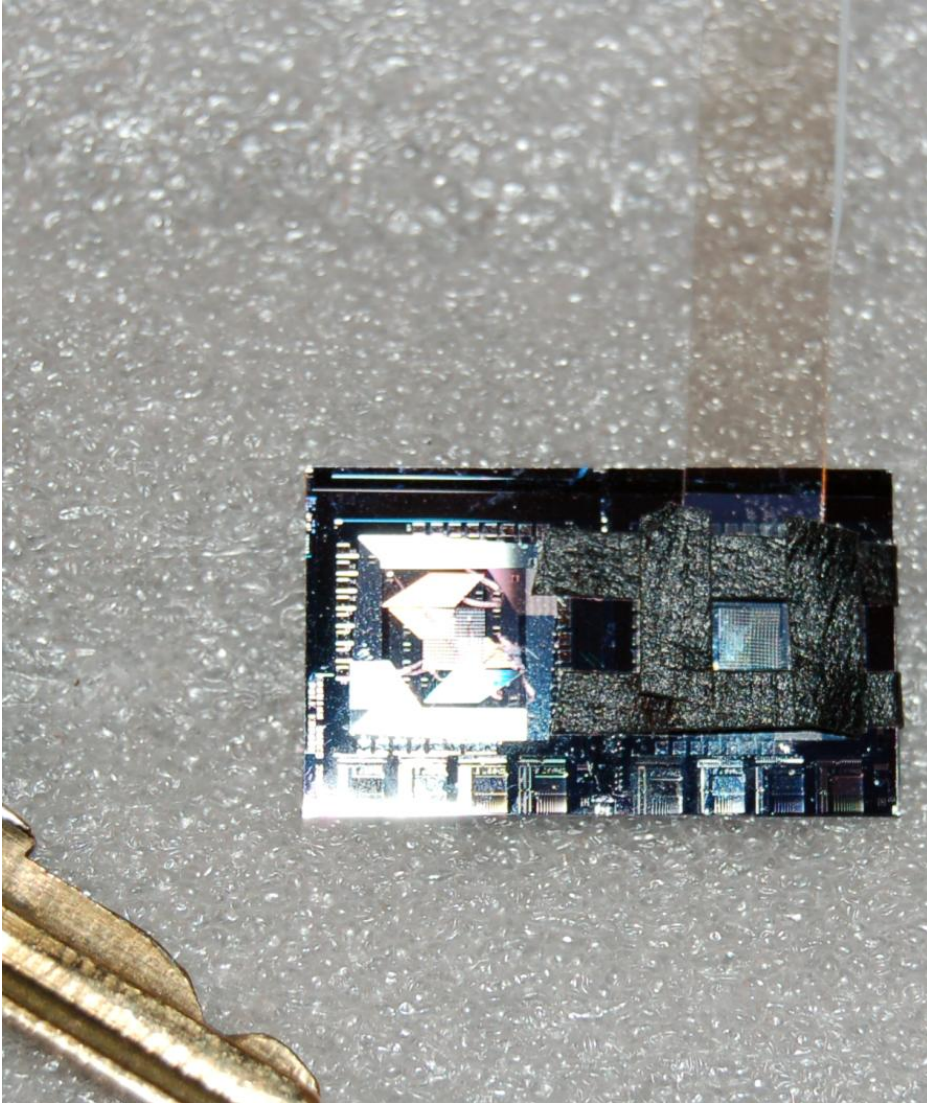


Figure 38: EOAM device with non-active area masked off by black tape.

b. Reflection simulation of film stack

The interferograms from the EOAM devices at varying voltage clearly show changes to the interference patterns. The interferograms are the result of constructive and destructive interference between the wavefront of the active device optical path and the wavefront of the non-changing optical path (reference wavefront) in the single arm interferometer system. Simulation code was written in Matlab® to explore the approximate reflectance of the film stack for the EOAM device and is included in

Appendix E, g5_nm_PDLC.fig and g5_nm_PDLC.m. The optical properties of the film stack for the EOAM device are input and the algorithm models the reflectance [19] at the various interfaces based on the layers of the film stack. Figure 39 shows the estimate for reflectance into air of the EOAM device versus thickness of the ITO layer for 541 nm illumination at 2 degrees angle of incidence as 0.84 to 0.91. This reflectance estimate includes the active device optical path and the wavefront from the non-changing optical path. Figure 40 gives the reflectance, 0 to 0.06, for the wavefront from the non-changing optical path that includes only the top four layers of the EOAM device.

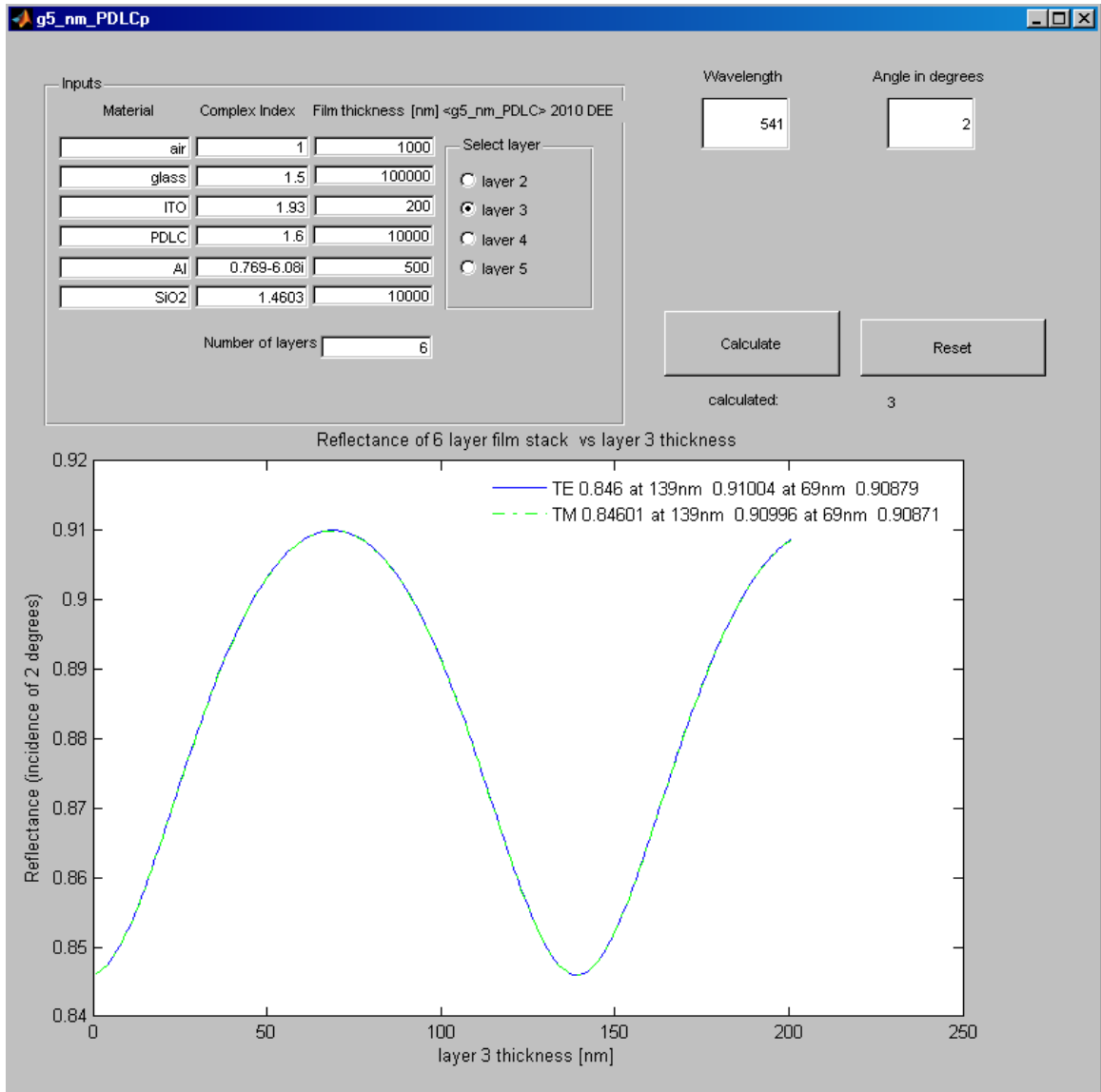


Figure 39: Modeled reflectance for EOAM device (entire film stack).

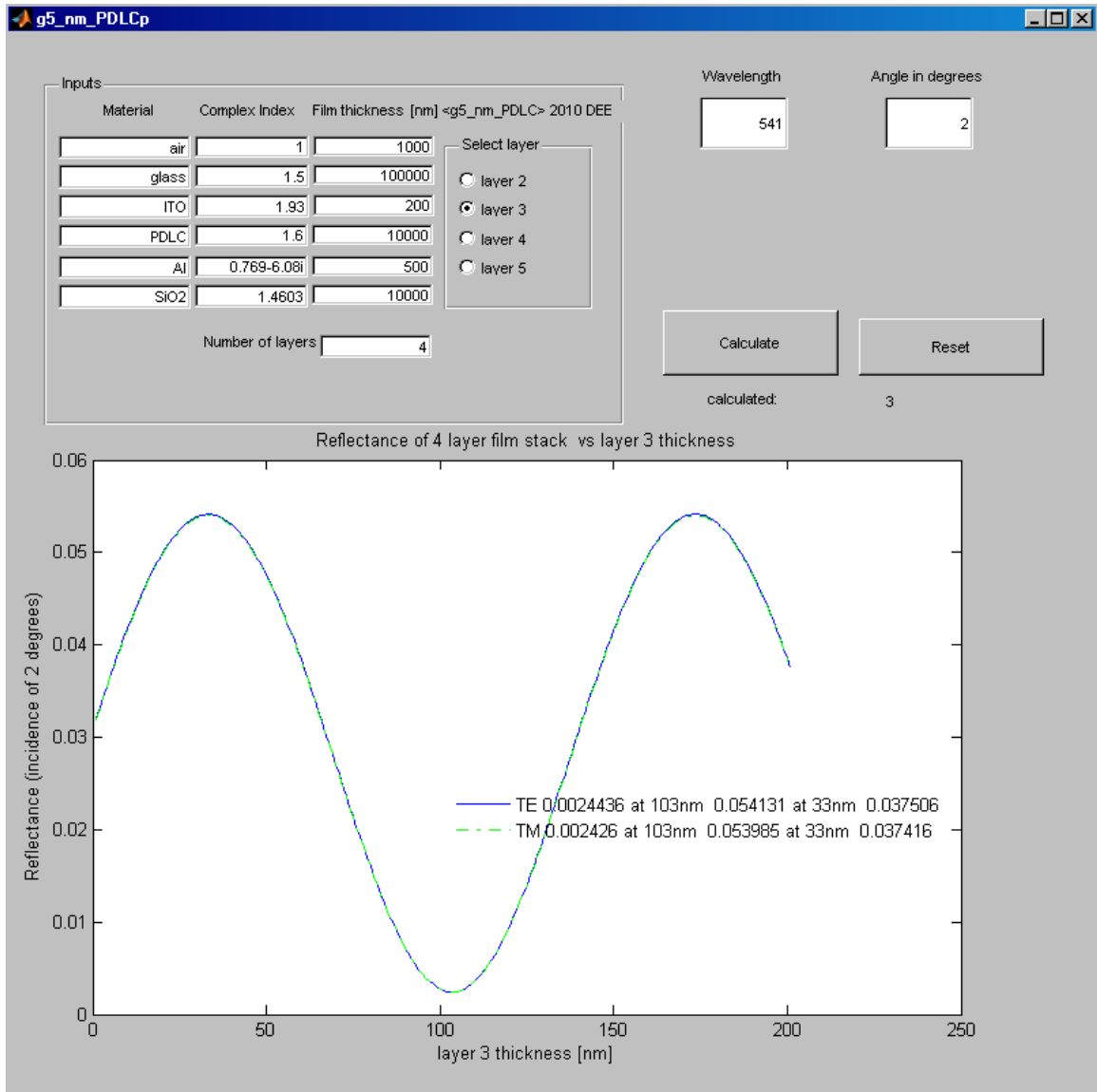


Figure 40: Modeled reflectance for top 4 layers of EOAM device.

c. Imaging theory for phase extraction

Extraction of the blind phase from the interferograms was necessary to characterize the EOAM as a function of the applied voltage. An iterative algorithm (Appendix F, xu_2007_ph_ext_05182010_data_p.m) has been implemented for blind phase extraction for characterization of an electro-optic adaptive microlens device. The

EOAM was designed to operate in the visible wavelengths and due to its reflective nature it was possible to collect interferograms of the reference wavefront and the object wavefront using a common optical path. It is not possible to measure the reference and the object wavefronts independently for the reflective device; and therefore the algorithm has been developed for this case. To validate the code and better understand its region of usefulness a series of simulations were completed to verify the algorithm based on assumptions.

Evaluation of optical surfaces is commonly accomplished via phase-shifting interferometry (PSI). PSI techniques have been used for more than forty years [28, 29]. All PSI techniques are based on multiple collections of the interference of a reference wavefront and an object wavefront at some point in space. An interferogram is a mapping of one of these collections of interference.

In general each interferogram, collected as an image, is a record of the constructive and destructive interference of the reference and object wavefronts at a plane for some finite interval of time. The image irradiance collected at each point (x,y) of the detector is given by:

$$I(x, y, \alpha) = (A_o(x, y))^2 + A_r^2 + 2A_o(x, y) A_r \cos[\varphi(x, y) - \alpha], \quad (3.4.1)$$

where $\varphi(x,y)$ is the relative phase of each image point and α is the relative phase difference between the reference and object wavefronts. $A_o(x,y)$ is the electric field of the object wavefront and A_r is the electric field of the reference wavefront. It is assumed that the reference wavefront is non-varying across the image.

Many of the modern methods of PSI require collection of multiple ($n \geq 3$) interferograms [30-36] at varying phase differences between the reference and object

wavefronts. The requirement for multiple interferograms is due to the mathematical methods used for extraction of information from the interferograms. Equation (3.4.1) has four unknowns and thus requires three or more images to develop a solution.

There are techniques that can derive the wavefront or extract the phase difference utilizing only two interferogram images [37, 38]; however these techniques require that the reference wavefront and the object wavefront be measurable independently of one another.

A method utilizing an iterative algorithm for blind phase extraction [39] allows for extraction of the phase without measurement of the reference wavefront. This method was used as the starting point for development of a technique to extract phase for an electro-optic micro device that functions in the visible and is also reflective.

The electro-optic micro device [27] was fabricated on a silicon substrate and allows for addressing of pixels. The device has an active layer of polymer dispersed liquid crystals (PDLC) that change alignment under an applied electric field. The alignment of the PDLC causes a change in the refractive index and thus alters the effective optical path length. The device is reflective and the incoming radiation makes a double pass through the PDLC before exiting.

To characterize the change in optical path length with applied electric field the device was put into a Twyman-Green (dual beam) interferometer that was assembled on an electrical probe station [25]. Upon measurement and calculation of the phase change of the micro device, with replication of the results it was found that the arms of the interferometer were not stable. The extracted phase in data collected over a period of 20 minutes varied by as much as π due to the ambient air flow, temperature, and humidity.

In the optical setup and alignment process it was also noted that reflections from other surfaces of the device allow for collection of interferometric fringes. These “nuisance” fringes are a result of the upper layers of the electro-optic micro device and are noise to the device output. However, this ”nuisance” wavefront can be used as a reference wavefront in a single beam interferometer system [27]. This reference wavefront cannot be measured independently of the object wavefront exiting the device.

A modified algorithm has been developed and tested. This new technique allows for relative phase extraction of the active layer of the micro device while utilizing the “nuisance” wavefront as the non-changing reference. This new technique is based on the blind phase shift extraction [39] technique previously published; however it requires a different set of assumptions., which are discussed in detail in the following section. The modified algorithm was tested via simulated interferograms and a comparison of extracted phase shifts to known inputs. Data is presented for extracted phase shifts from the reflective electro-optic micro device.

d. Assumptions and algorithm

The previously published iterative algorithm for blind phase extraction [39] combines the least square regression method and formulae that allow extraction of the unknown phase shift utilizing only the intensities of the two interferograms. While the algorithm requires far less measured or controlled input than other methods [28-38], it has a few stated assumptions and restrictions.

Equation (3.4.1) is the basis for the two required interferograms and are given as

$$I_1(x, y) = (A_o(x, y))^2 + A_r^2 + 2A_o(x, y) A_r \cos[\varphi(x, y)], \quad (3.4.2)$$

and

$$I_2(x,y) = (A_o(x,y))^2 + A_r^2 + 2A_o(x,y)A_r \cos[\varphi(x,y) - \alpha]. \quad (3.4.3)$$

$I_1(x,y)$ is given for $\alpha = 0$, where as $I_2(x,y)$ includes a change in phase equal to α . The algorithm [39] previously published requires that the following assumptions hold true:

- I. $A_o(x,y)$ and $\varphi(x,y)$ are the real amplitude and phase distributions of the object wave and the region of interest is large enough that the distribution of φ is random.
- II. A_r is the constant amplitude of a plane reference wave.
- III. The arbitrary phase shift of the reference wave between two images is α and $0 < \alpha < \pi$.
- IV. Inputs of $A_r > A_o$ maximum, as is the case in practice to guarantee correct recording.

As stated, the algorithm previously published works well over a wide range of phase shift from 0.4 to 2.5 radians [39].

In this work the use of the algorithm has been extended to the case for the reflective electro-optical adaptive micro-device. As such, the algorithm requires the two interferograms given in equation (3.4.2) and (3.4.3) as well as the following assumptions:

- i. $A_o(x,y)$ and $\varphi(x,y)$ are the real amplitude and phase distributions of the object wave and the region of interest is large enough that the distribution of φ is random.
- ii. A_r is the constant amplitude of an **unchanging** reference wave.
- iii. The arbitrary phase shift of the **object** wave between two images is α and $0 < \alpha < \pi$.
- iv. **Calculated A_r / A_o ratio** of greater than one.
- v. **Calculated values in the iterations of the least squares regression must remain real valued.**

This modified algorithm works well over a wider range of phase shift **depending on the calculated A_r / A_o ratio** as is shown in the Simulations section.

Assumption I and i are the same as both algorithms are designed to extract the phase change between the reference wavefront and an object wavefront based on equations (3.4.2) and (3.4.3).

Assumption II, III, and IV are required because the initial algorithm is designed to extract the object wavefront, which can be back propagated to calculate the amplitude and phase distribution of the object. For this work the amplitude of the object wave and reference wave are assumed to be non-changing as the object is shifted in phase by α between the two interferograms. Therefore assumptions II and III are changed to assumptions ii and iii.

Assumption iv is required due to the reflective nature of the electro-optic micro device. However, as stated in [29, 38] and inferred by many using PSI, wavefront reconstruction by two-step interferometry requires $A_r(x,y)$ to be chosen as a constant greater than the maximum of $A_o(x,y)$. This is due to the image recording method on silver halide film. Due to the non-linearity of the foot of the image transfer curve the beam ratio,

$$\mathbf{R} = \left(A_r(x,y) \right)^2 / \left(A_o(x,y) \right)^2, \quad (3.4.4)$$

is stated in [40] as requiring a minimum $\mathbf{R} > 1$. This requirement has been carried over for the use of digital image detectors, even though most digital detectors are linear down to much lower irradiances. For dual or multiple beam interferometers the wavefronts can be attenuated to ensure the beam ratio remains greater than one. Attenuation of the wavefronts independently is not possible with a single beam system as used with a reflective electro-optical adaptive micro-device [27].

Assumption v is applied as a result of using the algorithm on simulated and experimental data. Further explanations and examples follow.

Following the algorithm [39], using equations (3.4.2) and (3.4.3), the sum and the difference of the two interferograms are

$$I_2(x, y) - I_1(x, y) = 4A_o(x, y)A_r \sin[\varphi(x, y) - \alpha/2] \sin(\alpha/2), \quad (3.4.5)$$

and

$$I_1(x, y) + I_2(x, y) = 2(A_o(x, y))^2 + 2A_r^2 + 4A_o(x, y)A_r \cos[\varphi(x, y) - \alpha/2] \cos(\alpha/2). \quad (3.4.6)$$

Using assumption i and $\sin^2[\varphi(x, y) - \alpha/2] + \cos^2[\varphi(x, y) - \alpha/2] = 1$ with equations (3.4.5) and (3.4.6), the quadratic equation

$$I_o^2 - pI_o + q = 0 \quad (3.4.7)$$

is obtained, where $I_o = A_o^2$, $I_r = A_r^2$, $p = I_1 + I_2 + 2I_r \cos(\alpha)$, $q = [(I_1 + I_2 - 2I_r)^2 + (I_2 - I_1)^2 / \tan^2(\alpha/2)]/4$, and the coordinates (x, y) are omitted from A_o , I_o , I_1 , and I_2 . Solving for the real roots by assuring $(p^2 - 4q)$ and $p - (p^2 - 4q)^{1/2}$ are positive as in [39], the irradiance and electric field are found for the object wavefront,

$$I_o = \left[p - (p^2 - 4q)^{1/2} \right] / 2, \text{ and } A_o = \left\{ \left[p - (p^2 - 4q)^{1/2} \right] / 2 \right\}^{1/2}. \quad (3.4.8)$$

Here assumption iv is used; validation will be shown in the simulations.

Rearrangement of equation (3.4.2) gives

$$\cos(\varphi) = (I_1 - I_o - I_r) / (2A_r A_o), \quad (3.4.9)$$

and

$$\sin(\varphi) = (I_2 - I_1) / [2A_r A_o \sin(\alpha)] + (I_1 - I_o - I_r) \tan(\alpha/2) / (2A_r A_o) \quad (3.4.10)$$

is obtained by substitution of equation (9) into equation (5).

The object wavefront $O(x,y) = A_o(x,y) \exp(i\varphi(x,y))$ can be calculated with I_1 , I_2 , A_r and α using equations (3.4.8)-(3.4.10).

Least squares regression can be used to find A_r and α by rearrangement of equation (3.4.6) as

$$I_2 = I_0 + I_r + c_1 \cos \varphi + c_2 \sin \varphi, \quad (3.4.11)$$

where

$$c_1 = 2A_o A_r \cos \alpha, \text{ and } c_2 = 2A_o A_r \sin \alpha, \quad (3.4.12)$$

and the summations for equation (3.4.13) are taken for the $N \times N$ pixels of the interferograms. The matrix,

$$\begin{pmatrix} N^2 & \sum \cos \varphi & \sum \sin \varphi \\ \sum \cos \varphi & \sum \cos^2 \varphi & \sum \cos \varphi \sin \varphi \\ \sum \sin \varphi & \sum \cos \varphi \sin \varphi & \sum \sin^2 \varphi \end{pmatrix} \begin{pmatrix} I_r \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \sum (I_2 - I_0) \\ \sum (I_2 - I_0) \cos \varphi \\ \sum (I_2 - I_0) \sin \varphi \end{pmatrix} \quad (3.4.13)$$

can be solved for values of I_r , c_1 , and c_2 allowing calculation of the reference electric field A_r and the phase shift α as

$$A_r = I_r^{1/2}, \text{ and } \alpha = \tan^{-1}(c_2 / c_1). \quad (3.4.14)$$

Thus using I_2 , A_o and φ the blind phase shift α is extracted.

Because only the interferograms I_1 and I_2 are known, the initial value for α is chosen from 0.1 to 0.4 times π and the initial value for A_r is the square root of the average pixel irradiance of $I_1(x,y)$. With these initial values, A_o and φ are calculated using equations (3.4.8)-(3.4.10). The least squares regression is then calculated and new values for the reference electric field, A_r , and the phase shift, α , are obtained. This iterative process is

allowed to continue until the value for phase shift converges to a difference of less than 1×10^{-5} radians.

The flow of the algorithm is shown in Figure 41. In the process of executing the algorithm it is necessary to validate the calculated intermediate values to ensure that the assumptions hold true and that the intermediate values are real. These validation steps are also shown in Figure 41. The outputs at step 13 include the blind phase shift, α , that results from the change of the optical path length of the object wavefront.

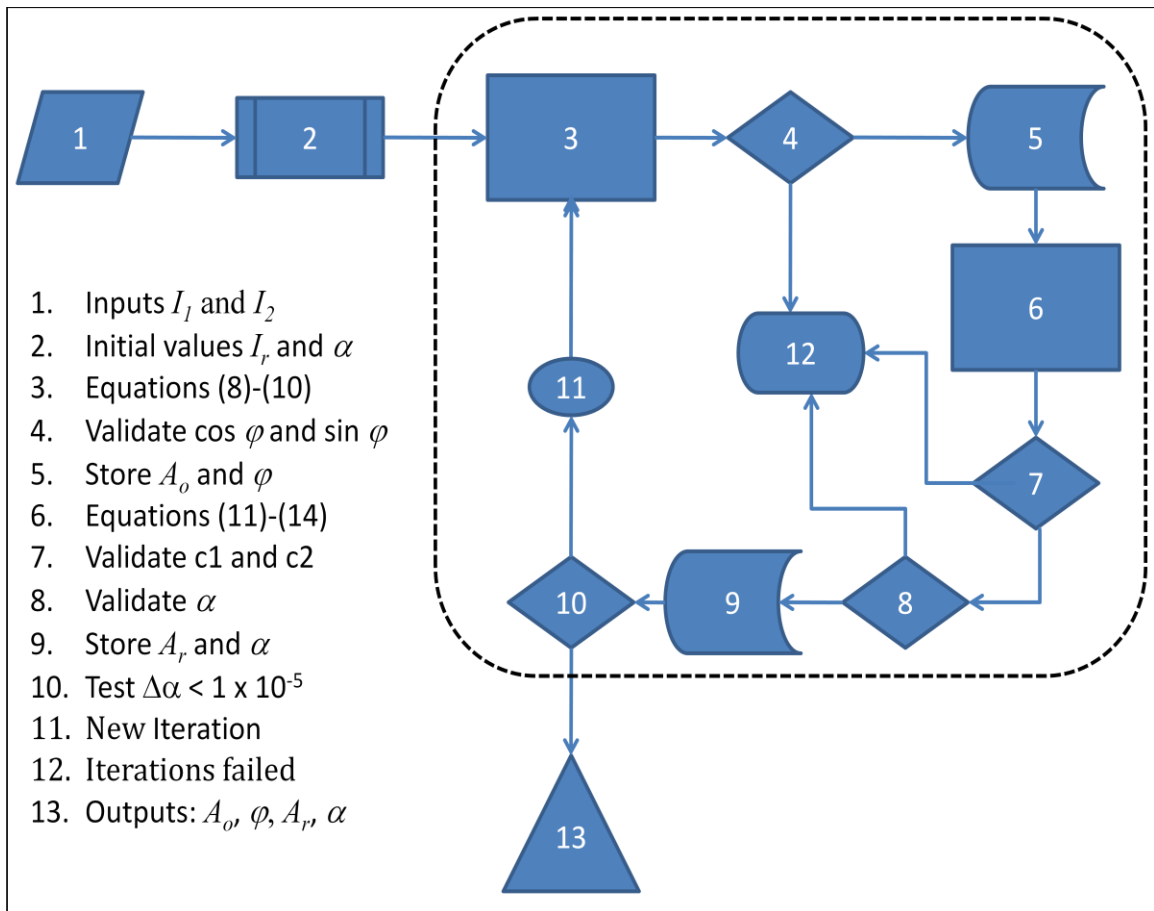


Figure 41: Flow diagram of algorithm showing its process and validation steps.

e. Simulations

The algorithm was coded in MATLAB® for use with RGB JPEG images captured on a Nikon D50 camera back. Upon coding and testing of the algorithm it was found that the

equations and iteration steps can lead to various non-real solutions. In these cases the resulting calculation of the phase (α) is not reliable. As such, the intermediate values for several of the variables are tested to validate assumption v.

It is important to remember that the variables of $A_o(x,y)$ and $\varphi(x,y)$ are functions of the pixel position in the wavefront. For equations (3.4.9) and (3.4.10) the calculated $\cos(\varphi(x,y))$ and $\sin(\varphi(x,y))$ must be real for all pixels in each iteration step. Then calculated value of $\varphi(x,y)$ will also be real. The least squares regression of the pixels data gives I_r , c_1 and c_2 that must be real, such that the calculated α from c_1 and c_2 will be real and in the range of 0 to π .

To validate the code and better understand its region of usefulness a series of simulations were completed. The region of interest for the simulations was driven by the need to extract phase from the electro-optic adaptive microlens. The algorithm was run to extract phase and appeared to function correctly for some cases, but failed to give reliable data for others.

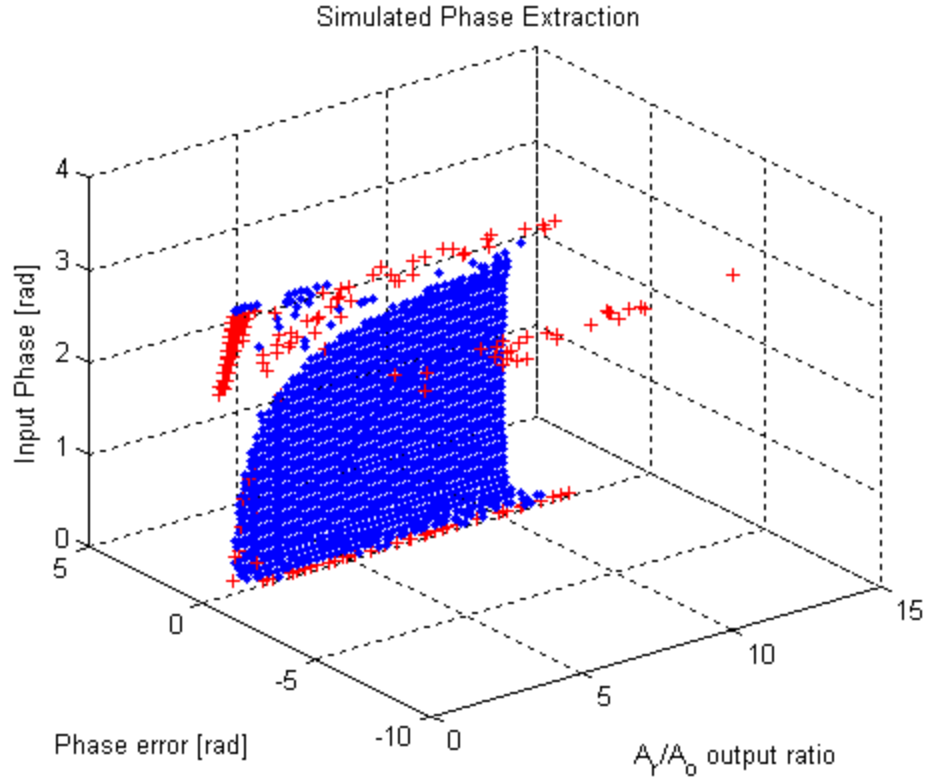


Figure 42: Phase extraction simulated with varying $A_r > A_o$ maximum.

Simulations of interferograms were generated using the following equations:

$$\varphi(x, y) = 0.6[(2\pi x/x_{\max})^2 + 3(2\pi y/y_{\max})], \quad (3.4.15)$$

$$I_1(x, y) = |A_o(x, y)\exp[i\{\varphi(x, y) + \theta + \rho_1(x, y)\}] + A_r\exp(i\theta)|^2, \quad (3.4.16)$$

and

$$I_2(x, y) = |A_o(x, y)\exp[i\{\varphi(x, y) + \theta + \rho_2(x, y) - \alpha\}] + A_r\exp(i\theta)|^2; \quad (3.4.17)$$

where x and y are the pixel indices, θ is the arbitrary phase as the wavefronts propagate, and $\rho(x, y)$ is random phase error. The distribution of phase for $\rho(x, y)$ used is normal with parameters of $\mu=0$ and $\sigma= 2\pi/100$. These interferograms were then used as inputs for phase extraction. Figure 42 shows output phase error for simulated data where the input A_r was set to 5 and the A_o value was incremented from 1 to 50. The simulated region over

this range of $A_r > A_o$ maximum clearly breaks assumption IV. However, as can be seen in the figure with varying A_r/A_o ratio, the calculated phase error is small over the range of calculated A_r/A_o ratio from 1 to 10. The plus symbol represents simulations where the value for α did not converge to the point where the change in alpha was less than 1×10^{-3} in less than 100 iterations. The algorithm transforms the inputs of A_r and A_o to force the outputs to result in an A_r/A_o ratio of one or greater. It thus allows for calculation of object and reference wavefronts with the object having a greater intensity than the reference. The usefulness of the algorithm for extracting phase is greatly increased by applying the requirement of assumption iv. This is the case needed for the electro-optic adaptive microlens.

As can be seen in Figure 43 the useful range of phase for calculated α is limited by the output A_r/A_o ratio. Only data with calculated α error of less than $2\pi/100$ are plotted; the dot symbols indicate input A_r/A_o ratio and the circle symbols indicate calculated output A_r/A_o ratio for simulations with the input phase. The useful range of α for the algorithm can also be seen in Figure 43. The circle symbols of output A_r/A_o ratio map the valid region for which the value of α can be extracted. This valid region for output A_r/A_o ratios greater than 5 includes a wider range of α , 0.1 to >2.5 radians, than the algorithm previously published [39].

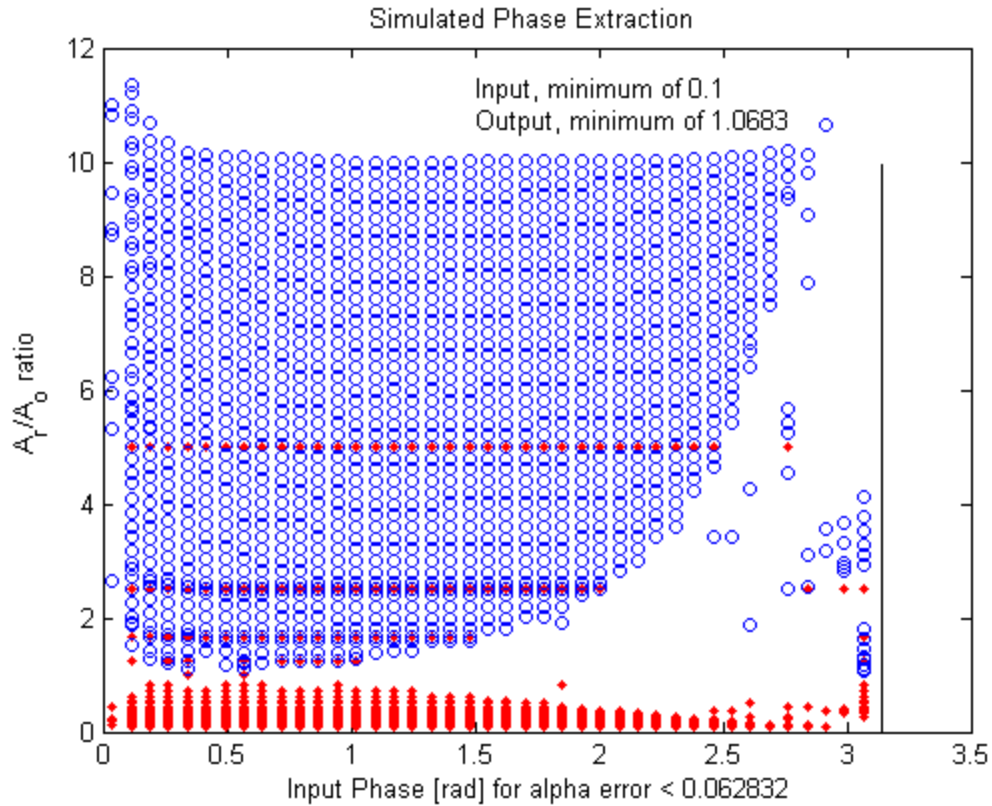


Figure 43: Phase with errors less than $2\pi/100$ comparing input and output A_r/A_o ratio.

f. Summary of phase extraction from devices

The electro-optic adaptive microlens was fabricated on a silicon substrate using standard microlithography techniques, with a film stack as shown in Figure 44. The aluminum layer is highly reflective, approximately 85-95%, and the upper layers of the film stack combine to give a reflectivity of approximately 5-15%. The reflectance from the upper layers is used as the reference wavefront in the phase extraction algorithm. The object path traverses the electro-optic material twice and is reflected from the aluminum interface. The electro-optic material used in the micro-device is polymer dispersed liquid crystals (PDLC) [15].

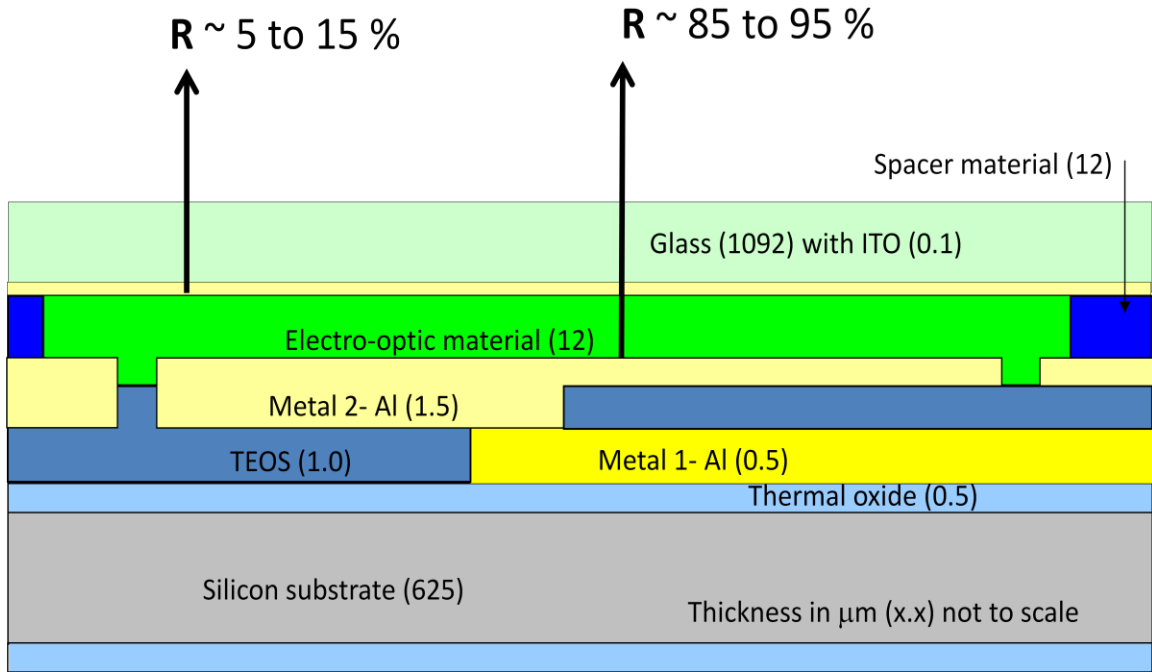


Figure 44: Layers of materials used in the reflective micro-device.

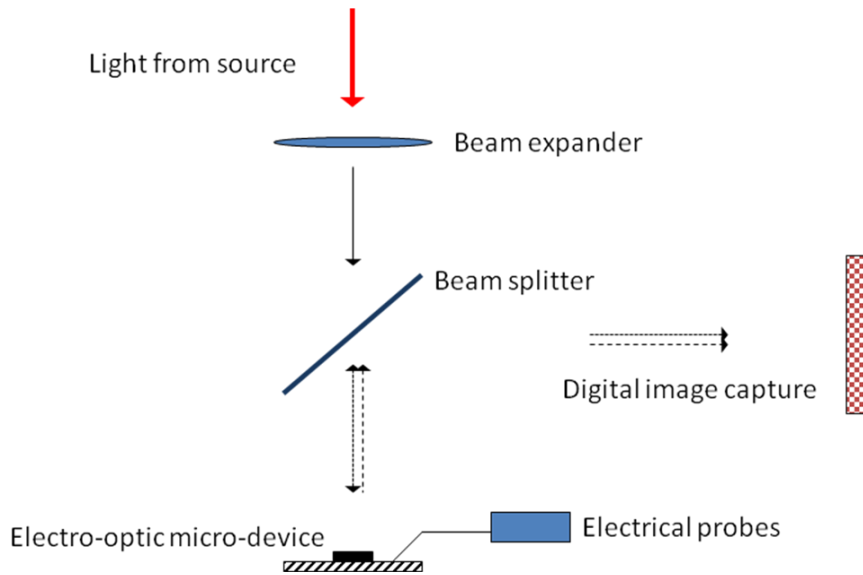


Figure 45: Diagram of single arm interferometer system used to collect interferograms.

Interferograms were collected from the micro-device utilizing an optical and electrical test system set up as a single arm interferometer[27]. As shown in Figure 45, the reference wavefront for the interferometer and the object wavefront travel the same path in the optical system. The object wavefront makes a double pass through the PDLC.

The refractive index of the PDLC is controlled by applying an electric field across the PDLC layer[15]. The top electrode is indium tin oxide (ITO) coated on the cover glass. The bottom electrode is polished aluminum of the metal 2 layer that has been patterned as pixels and is contacted to the metal 1 layer. The metal 1 layer is routed to the pads on the outer edge of the micro-device. The field is applied as 60 Hz AC from 3-240 volts through the probe pins.

Images were collected from a single arm interferometer of electro-optic adaptive microlens devices as the input voltage to the device was varied. Figure 46 shows two sample interferograms. The device itself is made up of an array of 16 by 16 pixels that are on a pitch of 160 micrometers. The device pixels were addressed with the same voltages for image capture. The images were then analyzed utilizing the algorithm with assumptions i to v to estimate the phase change due to applied field across the PDLC.

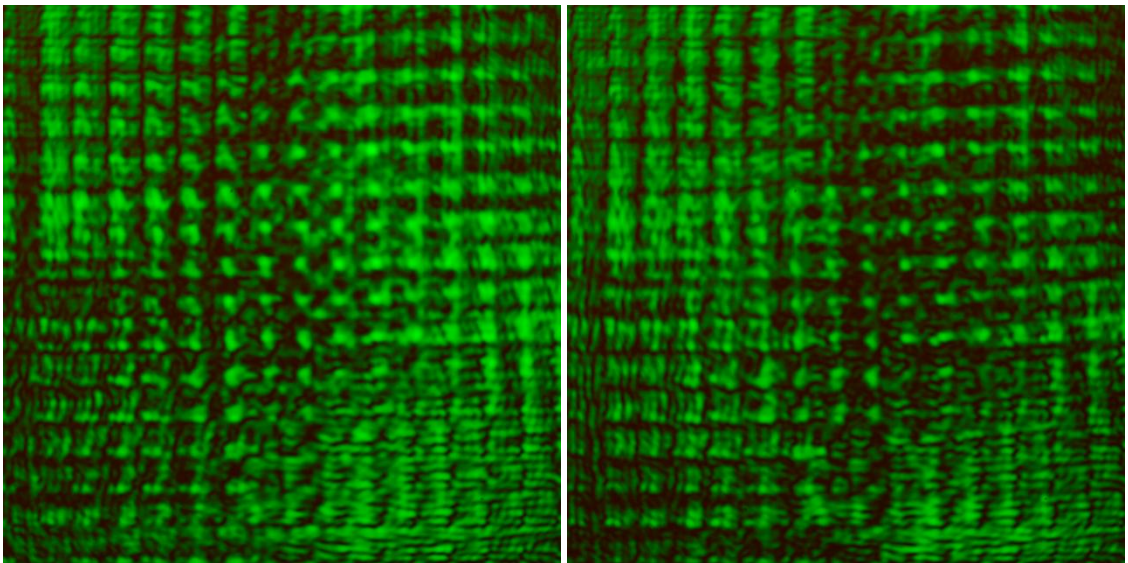


Figure 46: Interferograms of PDLC device at 3 volts (left) and 240 volts (right).

Figure 47 is a plot of repeated runs of the algorithm across nine 256x256 pixel regions of the interferograms. The plus symbols indicate that the value for α did not converge to the point where the change in alpha was less than 1×10^{-5} in less than 200 iterations and the x symbols indicate that the values for calculated $\cos(\varphi(x,y))$ or $\sin(\varphi(x,y))$ were complex so the iteration cycle was terminated. The diamond symbols represent α values where the change in alpha was less than 1×10^{-5} and the phase extraction was successful. The star symbol was plotted to show the average value of α for the successful iterations at each voltage. The variations in α at each voltage are due to the regions for which the algorithm was run (each region was 256 by 256 pixels of the interferograms). The calculated A_r/A_o ratio for this device was 2.35, which is in the range for valid extraction of α from approximately 0.1 to 2.0 rad as shown in Figure 43. The object wave has about 5.5 times the energy of the reference wave.

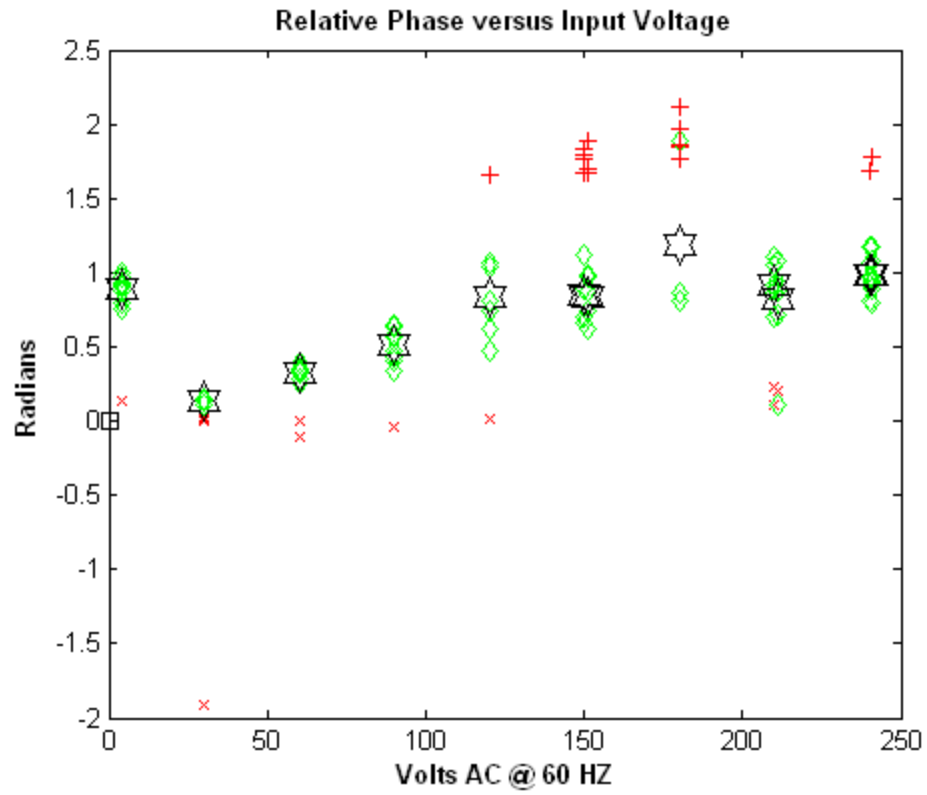


Figure 47: Sample phase extraction from interferograms of electro-optic adaptive microlens devices.

F. Summary of EOAM Results

Single cell transmission devices were used for initial characterization of the PDLC process and were evaluated based on double arm interferometer with two passes through the device

The electro-optic adaptive microlens (EOAM) system was designed and built. Three fabrication cycles were completed for the reflective device wafers. The wafers had multiple devices with 160 μm pixels in an 16x16 array. The devices were designed at two focal lengths and with 2 and 4 phase levels.

The EOAM devices from fabrication cycles 1 and 2 were evaluated based on double arm interferometer. This method of characterization of phase was found to be unstable.

The EOAM devices from fabrication cycles 2 and 3 were evaluated based on single arm interferometer. This method of characterization of phase was found to be stable. With applied voltage of 240 V AC, the EOAM devices were limited in phase change to less than 2 radians.

Code was written and utilized for the following four tasks:

1. Simulation of imaging device
 - a. setupworkspace160_3.m
 - b. lensf500bit_3.m
 - c. arrayfillbit_160.m
 - d. fPropfocal_160.m
 - e. address_fbit.m
2. Simulation of film stack
 - a. g5_nm_PDLC.fig
 - b. g5_nm_PDLC.m
 - c. multilayer_Guenther_interface5.m
 - d. multilayer_Guether_zzzface5_cycle_542.m

3. Blind phase extraction

- a. datain_phase_extract_file6.m
- b. gen_phase_ex_06102008.m
- c. simple_phase_ex_05162008.m
- d. singlearm_Iaverage_0802.m
- e. singlearm_Image_grab_03242010_plot.m
- f. xu_2007_ph_ext_020102010_data_p.m
- g. xu_2007_phase_extract_working10132008.m

4. Analysis of devices

- a. datareal_plotting_phase_BW_03242010.m
- b. datareal_plotting_phase_RGB_03242010.m
- c. plot_datareal_A.m
- d. s_strehl_02122009.m
- e. spot_strehl_09292009.m
- f. testA0Ar01152009.m

The objectives completed were:

1. Design of an EOA Microlens.
2. Modeling and simulation of a near field wave propagation system.
3. Build an adaptive optical element that is electronically controlled by addressing the 2-dimensional array of pixels.
4. Test and analyze the EOAM in a wave propagation system.

Three iterations of the EOAM devices with PDLC were built and 26 devices were optically tested. The first build had aluminum surface roughness leading to high scatter and the phase shift could not be quantified. The aluminum CMP process was implemented for the second and third builds. For optical device testing the phase shift was extracted using a newly developed method for blind phase extraction.

Problems related to PDLC material and process were:

1. 2π phase change not possible ($\Delta n \sim 0.03$, thickness $\sim 11 \mu\text{m}$)
2. shorting of device due to high drive voltage
3. Delaminating of device

Target $\Delta n \sim 0.2$, thickness $\sim 1.6 \mu\text{m}$

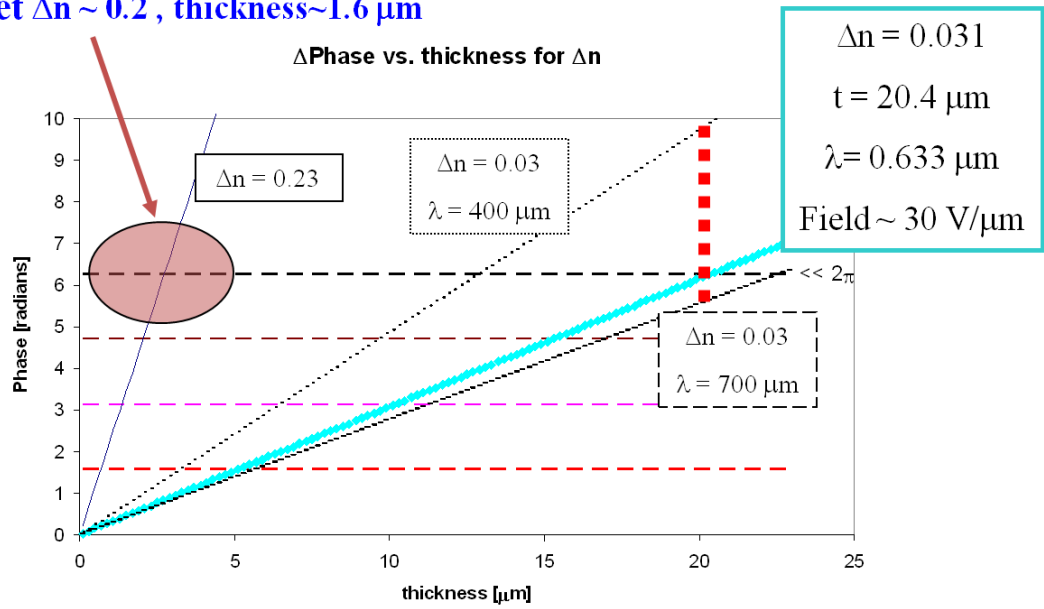


Figure 48: Delta phase versus electro-optic material thickness for Δn .

Need new electro-optic material with index change at low voltage ($\Delta n \sim 0.23$, thickness $\sim 1.6 \mu\text{m}$, $V < 10$ volts). Figure 48 shows the design space for the electro-optic material phase change based on equation (3.4.18). The red dotted area represents the PDLC used with $\Delta n \sim 0.03$ and a thickness needed of 20 micrometers.

Delta phase is calculated as:

$$\Delta\phi = \phi_2 - \phi_1 = (n_2 - n_1) z \frac{2\pi}{\lambda} = \Delta n z \frac{2\pi}{\lambda} \quad (3.4.18)$$

New process steps developed for the EOAM device fabrication were:

1. CMP of Metal 2 (patterned aluminum over via in TEOS to aluminum)
2. Patterning for Spacer layer (SU-8 negative i-line photoresist)
3. PDLC precursor mixture
4. PDLC fill and cover with ITO glass slide
5. UV exposure to cure PDLC and attach slide.

V. Conclusions

The goal of the present research was to demonstrate the viability of an electro-optic adaptive microlens (EOAM) system in imaging applications requiring broadband illumination in the visible region. Building an electro-optic adaptive microlens system utilizing polymer dispersed liquid crystals (PDLC) and pixel addressing was unique. The EOAM devices that were built modify the light path in the visible wavelengths; however, the designed lens effect was not realized because the PDLC change in phase for the film stack was limited to approximately 2 radians. Processes for design, fabrication, and testing of an electro-optic adaptive microlens (EOAM) system were however implemented. Software code was developed for design and simulation of the EOAM; and new fabrication processes for building reflective devices were developed and characterized. In addition, a new algorithm for blind phase extraction for characterization of an optical device that acts as a Pohl fringe producing system was developed, simulated and implemented in code. The development of a design model for the EOAM system and validating it with the images formed by a real electro-optic adaptive microlens system has provided the knowledge base needed for implementation of adaptive electro-optic lenses for the visible region, and, a process which can be used for further improvement of the microsystem. The model parameters can be adjusted for new electro-optic materials that may become available that do not have the limitations of PDLC.

The processes implemented here for design, fabrication, and testing of an electro-optic adaptive microlens (EOAM) system can be applied to a pixilated phase-only spatial light modulator (SLM). The pixel size, PDLC layer thickness, and resulting drive voltages can be adjusted and optimize to implement a system for relative phase changes

of less than one radian. Such a SLM would be useful element in optical systems needing small phase aberration corrections.

References

- [1] G. Vdovin, M. Loktev, and A. Naumov, "On the possibility of intraocular adaptive optics," *Opt. Express*, vol. 11, pp. 810-817, 2003.
- [2] G. Vdovin, M. Loktev, A. Simonov, V. Kijko, and S. Volkov, "Adaptive correction of human-eye aberrations in a subjective feedback loop," *Opt. Lett.*, vol. 30, pp. 795-797, 2005.
- [3] Rejean Munger, Linda E. Marchese and Lijan Hou, "Phase modulators for refractive corrections of human eyes", Proc. SPIE 5578, 251 (2004); doi:10.1117/12.567605.
- [4] D. Miller, L. Thibos, and X. Hong, "Requirements for segmented correctors for diffraction-limited performance in the human eye," *Opt. Express*, vol. 13, pp. 275-289, 2005.
- [5] R. D. Blum, D. P. Duston, W. Kokonaski, J. Thibodeau, Y. Katzman, and U. Efron, "Electro-optic lens with integrated components," U.S. Patent 6871951, 2005.
- [6] <http://www.kodak.com/global/en/corp/historyOfKodak/historyIntro.jhtml?pq-path=2687>, Mar. 17, 2005.
- [7] http://www.bell-labs.com/about/history/laser/laser_uses.html, Mar. 17, 2005.
- [8] H. Kawamoto, "The history of liquid-crystal displays," *Proceedings of the IEEE*, vol. 90, pp. 460-500, 2002.
- [9] P. F. McManamon, *et al.*, "Optical phased array technology," *Proceedings of the IEEE*, vol. 84, pp. 268-298, 1996.
- [10] T. Nagayasu, *et al.*, "A 14-in.-diagonal full-color a-Si TFT LCD," in *Display Research Conference, 1988., Conference Record of the 1988 International*, 1988, pp. 56-58.
- [11] M. Schadt and W. Helfrich, "VOLTAGE-DEPENDENT OPTICAL ACTIVITY OF A TWISTED NEMATIC LIQUID CRYSTAL," *Applied Physics Letters*, vol. 18, pp. 127-128, 1971.
- [12] A. Tanone, Z. Zhang, and C.-M. Uang, vol. 7, ed. *Microwave and Optical Technology Letters*, 1994, pp. 285-289.
- [13] D. P. Resler, D. S. Hobbs, R. C. Sharp, L. J. Friedman, and T. A. Dorschner, "High-efficiency liquid-crystal optical phased-array beam steering," *Opt. Lett.*, vol. 21, pp. 689-691, 1996.

- [14] Yun Hsing Fan, Hongwen Ren and Shin Tson Wu, "Electrically controlled lens and prism using nanoscale polymer-dispersed and polymer-networked liquid crystals", *Proc. SPIE* 5289, 63 (2004); doi:10.1117/12.526148.
- [15] H. Ren, Y.-H. Fan, and S.-T. Wu, "Tunable Fresnel lens using nanoscale polymer-dispersed liquid crystals," *Applied Physics Letters*, vol. 83, pp. 1515-1517, 2003.
- [16] X. Wang, H. Dai, and K. Xu, "Tunable reflective lens array based on liquid crystal on silicon," *Opt. Express*, vol. 13, pp. 352-357, 2005.
- [17] M. T. Gruneisen, L. F. DeSandre, J. R. Rotge, R. C. Dymale, and D. L. Lubin, "Programmable diffractive optics for wide-dynamic-range wavefront control using liquid-crystal spatial light modulators," *Optical Engineering*, vol. 43, pp. 1387-1393, 2004.
- [18] Abdul Ahad S. Awwal, Brian J. Bauman, Donald T. Gavel, Scot S. Olivier, Steve Jones, Dennis A. Silva, Joseph L. Hardy, Thomas B. Barnes and John S. Werner, "Characterization and operation of a liquid crystal adaptive optics phoropter", *Proc. SPIE* 5169, 104 (2003); doi:10.1117/12.510393.
- [19] R. D. Guenther, *Modern Optics*: John Wiley & Sons, 1990.
- [20] J. W. Goodman, *Introduction to Fourier Optics*, 2nd ed.: McGraw Hill Companies, Inc, 1996.
- [21] M. Born and E. Wolf, *Principles of Optics*, 7th ed.: Cambridge University Press, 1999.
- [22] E. Hecht, *Optics*, 4th ed.: Addison Wesley, 2002.
- [23] Marek Aleksander and Stanislaw J. Klosowicz, "Effect of preparation method on PDLC morphology and properties", *Proc. SPIE* 5565, 389 (2004); doi:10.1117/12.581217.
- [24] J. D. LeGrange, *et al.*, "Dependence of the electro-optical properties of polymer dispersed liquid crystals on the photopolymerization process," *Journal of Applied Physics*, vol. 81, pp. 5984-5991, 1997.
- [25] D. E. Ewbank, "Optical Test System for Reflective Electro-Optical Adaptive Micro-Device Phase Measurement," in *Frontiers in Optics*, OSA Technical Digest (CD) (Optical Society of America, 2006), paper OFMC6.
- [26] N. Nampalli, "Optimization of Chemical Mechanical Planarization (CMP) of Aluminum over Topography", RIT Undergraduate Research Symposium (2006), unpublished.

- [27] D. E. Ewbank, "Single Arm Interferometer System for Reflective Micro-Device Phase Measurement," in *Optical Fabrication and Testing*, OSA Technical Digest (CD) (Optical Society of America, 2010), paper OWC2.
- [28] D. Malacara, *Interferogram analysis for optical testing*, 2nd ed. vol. 84. Boca Raton, FL :: Taylor & Francis, 2005.
- [29] *Optical shop testing*, 3rd ed ed. Hoboken, N.J. :: Wiley-Interscience, 2007.
- [30] E. Luna, L. Salas, E. Sohn, E. Ruiz, J. M. Nunez, and J. Herrera, "Deterministic convergence in iterative phase shifting," *Applied Optics*, vol. 48, pp. 1494-1501, Mar 2009.
- [31] J. Xu, Q. Xu, and L. Chai, "Iterative algorithm for phase extraction from interferograms with random and spatially nonuniform phase shifts," *Appl. Opt.*, vol. 47, pp. 480-485, 2008.
- [32] X. F. Xu, L. Z. Cai, X. F. Meng, G. Y. Dong, and X. X. Shen, "Fast blind extraction of arbitrary unknown phase shifts by an iterative tangent approach in generalized phase-shifting interferometry," *Opt. Lett.*, vol. 31, pp. 1966-1968, 2006.
- [33] J. Vargas, N. Uribe-Patarroyo, J. A. Quiroga, A. Alvarez-Herrero, and T. Belenguier, "Optical inspection of liquid crystal variable retarder inhomogeneities," *Applied Optics*, vol. 49, pp. 568-574, Feb 2010.
- [34] L. Z. Cai, Q. Liu, and X. L. Yang, "Phase-shift extraction and wave-front reconstruction in phase-shifting interferometry with arbitrary phase steps," *Opt. Lett.*, vol. 28, pp. 1808-1810, 2003.
- [35] Z. Wang and B. Han, "Advanced iterative algorithm for phase extraction of randomly phase-shiftedinterferograms," *Opt. Lett.*, vol. 29, pp. 1671-1673, 2004.
- [36] P. Gao, B. Yao, N. Lindlein, K. Mantel, I. Harder, and E. Geist, "Phase-shift extraction for generalized phase-shifting interferometry," *Opt. Lett.*, vol. 34, pp. 3553-3555, 2009.
- [37] X. F. Xu, *et al.*, "Simple direct extraction of unknown phase shift and wavefront reconstruction in generalized phase-shifting interferometry: algorithm and experiments," *Opt. Lett.*, vol. 33, pp. 776-778, 2008.
- [38] X. F. Meng, *et al.*, "Wavefront reconstruction by two-step generalized phase-shifting interferometry," *Optics Communications*, vol. 281, pp. 5701-5705, 2008.
- [39] X. F. Xu, *et al.*, "Blind phase shift extraction and wavefront retrieval by two-frame phase-shifting interferometry with an unknown phase shift," *Optics Communications*, vol. 273, pp. 54-59, 2007.

[40] R. J. Collier, *Optical holography*. New York: Academic Press, 1971.

Appendix A: setupworkspace160_3.m, lensf500bit_3.m, arrayfillbit_160.m, fPropfocal_160.m, and address_fbit.m

```

function [N, FIG, LAMBDA, LZ, Q, FocaL, DIA, PIXEL, D0, X0, Y0] =
setupworkspace160_3(fig); % ****
% setup workspace for quartz >>> LAMBDA n k
%
% (for synthetic fused silica)
% LAMBDA [microns]      n          k      nvslope [delta n per V/micron]
% nd  .5875618         1.45846   0      .0015      is .030 per 20
% nF  .4861327         1.46313   0      .0015
% nC  .6562816         1.45637   0      .0015
%
% Vd = ( nd - 1 ) / ( nF - nC )
%
global N FIG LAMBDA LZ Q FocaL DIA PIXEL scale D0 X0 Y0 DF MAG show sh
zmag scrsz
start_clock = clock
set(0, 'Units', 'pixels');
scrsz = get(0, 'ScreenSize');
show=1; sh= 0;
N = 30
FIG = fig
LAMBDA = [0.5875618          1.45846          0          0.0015,
          0.4861327          1.46312          0          0.0015,
          0.6562816          1.45636          0          0.0015]

Vd = ( LAMBDA(1,2) - 1.0) / (LAMBDA(2,2)-LAMBDA(3,2))

%for lensf500
LZ = 20;          % lens thickness in microns
Q = 2             % quantize levels
zmag = 1.1        % times focal length for z1 value
FocaL = 150000;   % FocaL length in microns
PIXEL = 160       % PIXEL pitch in microns
if PIXEL > 16
    scale= PIXEL/16    % scale is microns per display "pixel units"
else
    scale = 1
end;
DIA = 368*scale;   % lens DIAMeter in microns
D0= 1              % input diameter for ef0 aperture
DF = 50;          % size of mask
X0=0              % input center for ef0 aperture
Y0=0              %input center for ef0 aperture

[f500bit, FIG] = lensf500bit_3(1);
[zbit, FIG] = arrayfillbit_160(f500bit,1);
dB = fPropfocal_160(zbit);
address_fbit(f500bit)
Q, PIXEL, D0, FocaL, MAG, scale
end_clock=clock
elapsed_time= etime(end_clock, start_clock)

```

```

function [f500bit, FIG] = lensf500bit_3(ask);
% build at 2D lens in complex array z with Q (Quantized levels of
phase)
%      zout = exp(i 2pi n LZ /LAMBDA * exp(-i pi r^2 / (LAMBDA FocaL) )
%              n index, LZ thickness, FocaL length of lens
%
%              DIA DIAMeter, PIXEL pitch in microns including
border
%
%USAGE run:
%      [N, FIG, LAMBDA, LZ, Q, FocaL, DIA, PIXEL, D0, X0, Y0] =
setupworkspaceNFIGLAMBDA_2(1);
%
%
%
global N FIG LAMBDA LZ Q FocaL DIA PIXEL D0 X0 Y0
close all

%LZ          % lens thickness in microns
%Q           % Quantize levels
%FocaL       % FocaL length in microns
%DIA         % lens DIAiameter in microns
%PIXEL       % PIXEL pitch in microns

N=floor(DIA/PIXEL)+3
if rem(N,2)==1
    N= N+1
end;
if ask == 1
maskname=strcat('lens','FocaL',num2str(round(FocaL)), 'N',num2str(N));

prompt = {'Enter matrix size: ', 'Enter mask name: ', 'Enter FocaL
length: ', ...
        'Enter DIAMeter: ', 'Enter Quantize levels: ', 'Enter x Offset:
', 'Enter y offset: ', ...
        'Enter PIXEL size: '};
title = 'Input for creating MASK for Imaging simulation';
lines= 1;
def = {num2str(N),maskname, ...

num2str(FocaL), num2str(DIA), num2str(Q), num2str(X0), num2str(Y0), num2str(
PIXEL)};
datas = inputdlg(prompt,title,lines,def);

datas
if isempty(datas)
    close(FIG);
    return;
end;
tic
N=str2num(datas{1});
maskname1=datas{2};
FocaL=str2num(datas{3});
DIA=str2num(datas{4});
Q=str2num(datas{5});
X0=str2num(datas{6});

```

```

Y0=str2num(datas{7});
PIXEL=str2num(datas{8});
end;
Xlens=-PIXEL/2
Ylens=-PIXEL/2

maskname=strcat('lens','DIA',num2str(round(DIA)), 'N', num2str(N));

r = DIA/2;
%start clock
%tic;
lx = ((N/2 +1)*PIXEL +Xlens)- r;
hx = ((N/2 -1)*PIXEL +Xlens)+ r;
ly = ((N/2 +1)*PIXEL +Ylens)- r;
hy = ((N/2 -1)*PIXEL +Ylens)+ r;
%   lx,hx,ly,hy

if lx < PIXEL | ly < PIXEL
    error('counter < N');
    else if hx > N*PIXEL | hy > N*PIXEL
        error('counter > N');
    end;
end;
%

x1 = (-N/2)*PIXEL
xh = ((N/2)-1)*PIXEL

[X,Y] = meshgrid(x1:(PIXEL):xh);

R = sqrt((X-Xlens).^2 + (Y-Ylens).^2) + eps;
z = ((sign(r-R))+1)/2.* exp(i * 2 * pi * LAMBDA(1,2) * LZ / LAMBDA(1,1)
)...
.* exp(-i * pi *R.^2 / (LAMBDA(1,1) * FocaL) );

%Quantizes the phase in Q levels

zmag= abs(z);
%angle(z)*180/pi; %fix for angle???????????
zang= ( round(angle(z)/(2*pi/Q)) ); %discretizes the phase
for m=1:N
for n=1:N
    if zang(n,m)< 0
        zang(n,m)= zang(n,m)+Q;
    end;
end;
end;
%zang
z= zmag.* exp(i.*zang.*(2*pi/Q));

if ask == 1
FIG=showproj(FIG+1,N,z,maskname);
%FIG=showproj(FIG+1,N,zang,'zang');
end;

```

```

%

% zout now in units of microns !!!!!!!
f500bit = zang;
assignin('base','f500bit',f500bit);
toc
return;

function [zbit, FIG] = arrayfillbit_160(arrayin, ask);
% build at 2D lens in complex array z with
%   zbit =
%           n rows   m columns
%
%           PIXEL size in microns
%
%USAGE run before this:
%       [f500, FIG] = lensf500_2(Q, 1);
%
%
%
%
global N FIG LAMBDA LZ Q FocaL DIA PIXEL scale D0 X0 Y0
N=512
%close all

%PIXEL = pin;      % pitch = pixel size plus border in microns
z = arrayin;
insize=size(z)

shn = N- PIXEL*insize(1)/scale
shm = N- PIXEL*insize(2)/scale;

if shn < 0 | shm < 0
    error('PIXEL*insize()/scale > N in arrayfillbit');
end;
%
%take z array and expand it to zbit array for PIXEL size of PIXEL
%
zbit=zeros(N);

Pscale = PIXEL/scale
temp= ones(Pscale);
for m=1:insize(2)
for n=1:insize(1)

    temp=temp*z(n,m);
zbit(Pscale*n-(Pscale-1):Pscale*n,Pscale*m-(Pscale-1):Pscale*m)=temp(1:Pscale,1:Pscale);

temp= ones(Pscale);
end;
end;
% fill in linear estimate for border on 1 micron

```



```

for m = Pscale:Pscale:insize(2)*Pscale
    for n = 1:insize(1)*Pscale
        zbit(n,m) = (zbit(n,m)+zbit(n,m+1))/2;
    end;
end;
for n = Pscale:Pscale:insize(1)*Pscale
    for m = 1:insize(2)*Pscale
        zbit(n,m) = (zbit(n,m)+zbit(n+1,m))/2;
    end;
end;
% zbit now in units of microns !!!!!!!

%shift to center at N/2 +1

zbit = circshift(zbit, [floor(shn/2)+1 floor(shm/2)+1]);

if ask == 1
    maskname=strcat('zbit','scale',num2str(scale));
    FIG=showproj(FIG+1,N,zbit,maskname);
    % FIG=showproj(FIG+1,N,angle(zbit),'angle zbit');
end;

return;

function dB = fPropfocal_160(bitdata); % ****
% Propagate an input illumination aperture e-field ef0, distance z1
% to an EOAM eoam1 in complex array of size N,
% then Propagate distance z2 to ouput e-field ef2,
%
% and plot
%
%USAGE run before this:
% [zfill, FIG] = arrayfillbit_3(f500,1);
%
%
global N FIG LAMBDA LZ Q FocaL DIA PIXEL scale D0 X0 Y0 z1 z2 show sh
zmag
global NT levels gen0 sizSelCh maxelN
global ef0 eoam1 eoami ef2 Prop1 Prop2 MAG

gen0 = 0; NT =1 % globals for testing?????????
close all

%N=30; For testing
%eoami = angle(data)./(2*pi/Q); % in bit units from -(Q/2)+1 to Q/2
%min(min(eoami))
%max(max(eoami))
%zmag = 2;
slamb = size(LAMBDA,1)
LZphase = zeros(slamb,1);
ephase = zeros(N, N, slamb, 'double');
ephase(:, :, 1) = angle(zmag * exp(i*bitdata*(2*pi/Q)));
if slamb>1
    LZphase(1) = mod((LAMBDA(1,2)*LZ*2*pi/LAMBDA(1,1)), 2*pi);

```

```

Vpm = ((ephase(:, :, 1) * LAMBDA(1, 1) / (LZ * 2 * pi)) / LAMBDA(1, 4));
for m = 2:slamb
    LZphase(m) = mod((LAMBDA(m, 2) * LZ * 2 * pi / LAMBDA(m, 1)), 2 * pi);
    ephase(:, :, m) = (LZphase(m) - LZphase(1) +
(LAMBDA(m, 4) * LZ * 2 * pi / LAMBDA(m, 1)) * Vpm);
end;
end;

%NT1= sizSelCh;
if gen0 == 0
    NT1 = NT;
end;

dB=zeros(NT1,1);
%show=1; sh= 0; FIG=0;
%    LAMBDA = [0.6328 1.5 0]; % [wavelength n k]
n1 = 1.000; % index of air for z1
n2 = 1.000; % index of medium for z2
%NA= % NA for object side of system
%Q = % total system OPL in microns

%D0= 30; X0=0; Y0=0; %input diameter and center for ef0
aperture
d1= 256; x1= 0; y1= 0; %input diameter and center for eoam1
aperture
d2= 300; x2=0; y2= 0; %input diameter and center for ef2
aperture

z0 = 0
z1= zmag*FocaL % (d1/2)/NA for only filled eoam1 %calculated z
distances
z2 = 1/(1/FocaL -1/z1)

dlens_d=zeros(N,N,3, 'double');

if gen0 == 0

dlens_d(:, :, :) = 1.0 * exp(i * ephase(:, :, :)); %lens in complex
array

FIG=showproj3(FIG+1,N,dlens_d, 'dlens_d');
%FIG=showproj(FIG+1,N,dlens_d(:, :, 2), 'dlens_F');
%FIG=showproj(FIG+1,N,dlens_d(:, :, 3), 'dlens_C');

[e0, FIG] = rect(1, D0, X0, Y0, sh); % aperture at z0
[e1, FIG] = rect(1, d1, x1, y1, sh); % aperture at z1
[e2, FIG] = rect(1, d2, x2, y2, sh); % aperture at z2

[Prop1, FIG] = chirp_Pscale(1,z1,sh); %use upchirp for fresnel
propagation kernel
[Prop2, FIG] = chirp_Pscale(1,z2,sh); %use upchirp for fresnel
propagation kernel

```

```

assignin('base','e0',e0);           % puts arrays into workspace
assignin('base','e1',e1);
assignin('base','e2',e2);
assignin('base','Prop1',Prop1);
assignin('base','Prop2',Prop2);
assignin('base','dlens_d',dlens_d);

ef0 = e0;           % phase assumed constant?????
ef1 = ncconv3(ef0,Prop1);           % fresnel Propagation --normalized
power

%else
% disp ( [' in loop with globals used'] );

end;

for b = 1 : NT1

    dBx=99999;
    % [ran1,ran2]= vect2smatrix(data(b,:));
    % fixN = (N+maxelN)/2 +maxelN/2;
    % ran1(fixN+2*y1,fixN+2*x1)=0;
    % ran2(fixN+2*y2,fixN+2*x2)=0;
    % ran1=fftshift(ran1);
    % ran2=fftshift(ran2);
    %ran1(N,N)=0;
    %ran1a(1:N,1:N)=ran1(1:N,1:N);
    %ran2(N,N)=0;
    %ran2a(1:N,1:N)=ran2(1:N,1:N);

    eoam_d= ncmult3(e1,dlens_d);

    ef1e1= ncmult3(ef1,eoam_d);

    % normalize power to 1 at EOAM
    %for m=1:slamb
    %norm(m)=sum(sum(ef1e1(:, :, m).*conj(ef1e1(:, :, m))))           % Power = 1
    of wave front amplitude
    %ef1e1(:, :, m) = ef1e1(:, :, m)./(norm(m)^0.5);
    %end;

    pow1=sum(sum(ef1e1.*conj(ef1e1)))

    %f2 = (1/(i*LAMBDA*z2))*exp(-i*2*pi*z2/LAMBDA).*ncconv2(f1e1,Prop2);
    ef2 = ncconv3(ef1e1,Prop2);           % fresnel propagation --normalized
    power
    ef2e2 = ncmult3(e2,ef2);
    zI3 = ncSUMintensity3(ef2e2);

```

```

ef22 = ncconv3(ef2,Prop2);           % fresnel propagation --normalized
power

%ef222 = ncconv2(ef22,Prop2);       % fresnel propagation --normalized
power

%ef2222 = ncconv2(ef222,Prop2);     % fresnel propagation --
normalized power
%ef2222e2= (ef2222).*e2;

assignin('base','ef2e2',ef2e2);
assignin('base','zI3',zI3);

pow2=sum(sum(ef2e2.*conj(ef2e2)))

%count = count+1;

totalpow= pow1.*pow2
%disp ( [trialI triale3 pow1 pow2 pow3 totalpow] );
dBx= -10*log10(totalpow./pow1)

%dB(:, :, :, b)= dBx

if show ==1
    maskname=strcat('fProp3', ' N', num2str(N));

    masknamef0 =strcat(maskname, ' ef0' );
    masknamef1 =strcat(maskname, ' ef1' );
    masknamef1e1 =strcat(maskname, ' ef1e1' );
    masknamef2 =strcat(maskname, ' ef2 z2' );
    masknamef22 =strcat(maskname, ' ef2 pastz2' );
    masknamef2e2 =strcat(maskname, ' ef2e2' );
    masknameoam_d =strcat(maskname, ' eoam' );
    %masknamef2222 =strcat(maskname, ' ef24z2' );
    %masknamef2222e2 =strcat(maskname, ' ef24z2e2' );

    FIG=showproj(FIG+1,N,ef0, masknamef0 );
    FIG=showproj3(FIG+1,N,ef1, masknamef1 );
    FIG=showproj3(FIG+1,N,ef1e1, masknamef1e1 );
    FIG=showproj3(FIG+1,N,ef2, masknamef2 );
    % FIG=showproj3(FIG+1,N,ef2e2, masknamef2e2 );
    % FIG=showproj3(FIG+1,N,ef22, masknamef22 );
    FIG=showproj_color3(FIG+1,zI3, masknamef2e2 );
    %FIG=showproj(FIG+1,N,ef2222, masknamef2222 );
    %FIG=showproj(FIG+1,N,ef2222e2, masknamef2222e2 );
    %FIG=showplot(FIG+1,N,ef2e2, masknamef2e2 );
end;

%output info on lens system
so = z1
si = 1/(1/FocaL -1/so)
z2

```

```

MAG= -si/so

convolwithinputscale(zI3);

end;
return;

function address_fbit(fbit);
% calculate the number of pixel radii that are distinct
%
%
%
global N FIG add0 add1 add2 add3 PIXEL scrsz
maskname='address_fbit';
ask=1;imshowmag=3000;
fbitsize=size(fbit)
N=fbitsize(1)
znor=fbit;
zN=znor(N/2+1:N,N/2+1:N);
zNv(1:(N/2)^2)=zN;
%zsort=sort(zv);
zmin=min(zNv);
%counter=1000001
%while min(zv)<100
%   zmin = min(zv);
%for i=1:(N/2)^2
%   if zv(i) == zmin
%       zv(i)=counter;
%   end;end;
%   counter= counter+1;
%end;
%zv=zv-1000000
levelmax = max(zNv)

if ask == 1
    font=12;
larray = length(zN);
figure(FIG+1);
set(FIG+1,'Name',['Display of ',maskname]);
set(FIG+1,'Position',[scrsz(3)*.5 scrsz(4)*.53 scrsz(3)*.4
scrsz(4)*.4])
imshow(zN,[],'InitialMagnification',imshowmag)
text(0,0,['Real Part of
',maskname'],'FontSize',font,'VerticalAlignment','bottom');
zmin = num2str(min(min(real(zN))), '%+6.4g');
zmax = num2str(max(max(real(zN))), '%+6.4g');
text(0,larray,['min=',zmin,' max=',zmax'],'FontSize',font-
4,'VerticalAlignment','top');
figure(FIG+2);
set(FIG+2,'Position',[scrsz(3)*.5 scrsz(4)*.53 scrsz(3)*.4
scrsz(4)*.4])
plot(zNv,':bd')

```

```

end;

for i = 1:N/2
    zNN(1:N/2, i)=zNv(((i-1)*(N/2)+1):(i*(N/2)));
end;
zNN
z88=zNN(1:8,1:8)

if ask == 1
    figure(FIG+3);
    set(FIG+3, 'Position', [scrsz(3)*.5 scrsz(4)*.53 scrsz(3)*.4
scrsz(4)*.4])
    plot(z88, ':bd')
    font=12;
    larray = length(zN);
    figure(FIG+4);
    set(FIG+4, 'Name', ['Display of z88']);
    set(FIG+4, 'Position', [scrsz(3)*.5 scrsz(4)*.53 scrsz(3)*.4
scrsz(4)*.4])
    imshow(z88, [], 'InitialMagnification', imshowmag)
    text(0,0, ['Real Part of
', maskname], 'FontSize', font, 'VerticalAlignment', 'bottom');
    zmin = num2str(min(min(real(z88))), '%+6.4g');
    zmax = num2str(max(max(real(z88))), '%+6.4g');
    text(0, larray, ['min=', zmin, ' max=', zmax], 'FontSize', font-
4, 'VerticalAlignment', 'top');
end;
add0= find(z88 ==0)
add1= find(z88 ==1)
add2= find(z88 ==2)
add3= find(z88 ==3)

assignin('base', 'add0', add0);
assignin('base', 'add1', add1);
assignin('base', 'add2', add2);
assignin('base', 'add3', add3);
assignin('base', 'levelmax', levelmax);
assignin('base', 'z88', z88);
%toc
return;

```

Appendix B: datain_rerun_singlefile.m

```
%function datain_rerun31_singlefile.m
%Dale Ewbank 10/26/2005
% select files and assign voltage and color for Phase vs Voltage
analysis
%
rerunning = 1 % 1 for rerun with data in workspace, 0 for new
prompt = {'Running data files(0 for NEW, 1 for rerun, 2 for single
file):'}

    title = ['Input datain_rerun31.m'];
    lines= 1;
    def = {num2str(rerunning)};
    datas = inputdlg(prompt,title,lines,def);
    if isempty(datas)
        close all;
        disp(['User canceled datain_rerun31_singlefile.m'])
        return;
    end;
    rerunning =str2num(datas{1});

    if rerunning == 2
        prompt = {'ReRunning file#: '}
        title = ['Input datain_rerun31_singlefile.m'];
        lines= 1;
        def = {'99'};
        datas = inputdlg(prompt,title,lines,def);
        if isempty(datas)
            close all;
            disp(['User canceled datain_rerun31.m'])
            return;
        end;
        Nsingle =str2num(datas{1});
    end;
if rerunning == 0
    clear; rerunning =0;
end;
close all hidden;
imtool close all;
global Nsample fig
tic

hsize = 64;
fig =2;
fig_reuseask=0;
fig_reuse2ask=1;
start_sample=1;
Nsample = 70;
N=1 % number of "samples" to extract per image
if rerunning == 1 |rerunning == 2
    N = 3 % number of "samples" to extract per image
    Ntemp=size(sNmins);
    Nsample =Ntemp(1)
    clear temp1 temp2 temp3 rad_pix12 rad_pix31
end;
sample = 1; ss12 = 1; ss31 = 1;
```

```

set(0, 'Units', 'pixels')
scrsz = get(0, 'ScreenSize')

while sample < Nsample & rerunning == 0
[filename, pathname]=uigetfile({'*.tif'; '*.jpg'; '*.bmp'; '*.*'}, ['Select
Image > ', num2str(sample)]);
    if isequal(filename,0)
        disp('User selected Cancel. Ending file selection.')
        Nsample = sample -1
    else
        temp_path= pathname;
        disp(['User selected', fullfile(temp_path, filename)])
        sfile{sample}=filename;
        sample = sample +1;
    end;
end;

wave=0.5875618; % 0.4861327 for blue, 0.6562816 for red, 0.5875618
for green, 999 for all
    if wave == 0.4861327
        color = 3;
        colorr = 'b';
    elseif wave == 0.5875618
        color = 2;
        colorr = 'g';
    elseif wave == 0.6562816
        color = 1;
        colorr = 'r';
    else
        wave = 999;
        color = 3 ;
        colorr = 'b';
    end;
% test read to find files size and setup variables
if rerunning == 0
    moon1 = imread(fullfile(temp_path, sfile{Nsample}));
    [rmax, cmax, ncolor] = size(moon1)
    r2use = rmax/2-(round(N/2)*hsize)
    svolt = zeros(Nsample,1);
    scolorr{1}=colorr
    sNmins=ones(Nsample,1); sNminst= sNmins(1);
    xx1=ones(Nsample,1)*(hsize); xx1t=xx1(1);
    xx6=ones(Nsample,1)*(cmax-xx1(1)); xx6t=xx6(1);
end;

%create filter for smoothing
    sigma = 0.5*hsize;
    h2 = fspecial('gaussian',[hsize hsize/4],sigma);
if rerunning == 0
    svoltt=0;
    for sample = start_sample : Nsample

        prompt = {'Enter Voltage: ', 'Enter color> b g or r: ', ...
                };
        title = ['Input ', sfile{sample}];
        lines= 1;

```



```

        def      = {num2str(svoltt),colorr};
        datas   = inputdlg(prompt,title,lines,def);
        if isempty(datas)
            close(FIG);
            return;
        end;
        svolt(sample)=str2num(datas{1});
        scolorr{sample}=datas{2};
        colorr=datas{2};
        if sample > 1
            svoltt= svolt(sample)+(svolt(sample)-svolt(sample-1));
        end;
    end;
end;
if rerunning == 2
    start_sample= Nsingle
    Nsample=Nsingle
end;
for sample = start_sample : Nsample

    rad_pix12(sample)=0;
    rad_pix31(sample)=0;
    moon1 = imread(fullfile(temp_path, sfile{sample}));
    switch scolorr{sample}
        case 'b'
            wave = 0.4861327
            color = 3;colorr='b';
        case 'g'
            wave = 0.5875618
            color = 2;colorr='g';
        case 'r'
            wave = 0.6562816
            color = 1;colorr='r';
        otherwise
            disp(' Error on color input, please restart')
            return;
    end;
% pull region of image for finding fringe minimums
[x2,y2,I2,rect2] = imcrop(moon1,[1 r2use-hsize/2 cmax N*hsize]);
%imtool(h2, []);

%imtool(I2f, []);

if fig_reuseask == 0
    fig_reuse = 1;
    figure(fig_reuse);
    set(fig_reuse, 'Position', [1 scrsz(4)*0.5 scrsz(3)*.8
scrsz(4)*0.40]);
else
    fig= fig+1;
    fig_reuse=fig;
end;
figure(fig_reuse);
clf('reset');
set(fig_reuse, 'Position', [1 scrsz(4)*0.5 scrsz(3)*.8
scrsz(4)*0.40]);

```

```

    set(fig_reuse, 'Name', ['#', num2str(sample), ' of ', num2str(Nsample), '
', sfile{sample}, ' for minimums']);
    for n = 1 : N
        plot(I2(hsize/2+(n-1)*hsize, :, color), colorr); hold all;
    end;

    %find minimums between xx5 xx6
if rerunning == 0
    prompt = {'How many minimums? ', 'Enter xx1 for left of minimums:
', ...
            'Enter xx6 for right of minimums: '};
    title = ['Input ', sfile{sample}, 'for ', scolorr{sample}];
    lines= 1;
    def = {num2str(sNminst), num2str(xx1t), num2str(xx6t)};
    datas = inputdlg(prompt, title, lines, def);
    if isempty(datas)
        disp('NO or bad DATA');
        return;
    end;
    sNmins(sample)=str2num(datas{1});
    xx1(sample)=str2num(datas{2});
    xx6(sample)=str2num(datas{3});
    sNminst=sNmins(sample);
    xx1t=xx1(sample);
    xx6t=xx6(sample);
end;
if rerunning == 1 | rerunning == 2
    I2f = imfilter(I2, h2);
    xxrange=[xx1(sample):xx6(sample)];
    I2fm= ones(N*hsize, cmax, ncolor)*999;

    if fig_reuse2ask == 0
        fig_reuse2 = 2;
        figure(fig_reuse2);
        set(fig_reuse2, 'Position', [1 scrsz(4)*0.03 scrsz(3)*.8
scrsz(4)*0.40]);
    else
        fig=fig+1;
        fig_reuse2=fig;
    end;
    figure(fig_reuse2);
    clf('reset');
    set(fig_reuse2, 'Position', [1 scrsz(4)*0.03 scrsz(3)*.8
scrsz(4)*0.40]);
    set(fig_reuse2, 'Name', ['filter/marker #', num2str(sample), ' of
', num2str(Nsample), ' ', sfile{sample}, ' for minimums']);

    for n = 1 : N
        marker = imextendedmin(I2f(hsize/2+(n-
1)*hsize, xxrange, color), hsize*N);
        plot(I2f(hsize/2+(n-1)*hsize, :, color), colorr); hold all;
        I2fm(hsize/2+(n-1)*hsize, xxrange, color) =
imimposemin(I2f(hsize/2+(n-1)*hsize, xxrange, color), marker);
        plot(I2fm(hsize/2+(n-1)*hsize, :, color), 'k'); hold all;

```

```

ind1 = find(I2fm(hsize/2+(n-1)*hsize,xxrange,color) == 0);
K=size(ind1);
xmin=zeros(3,1)
switch sNmins(sample) % 3 used for rad_pix31, and is ahead of
first minimum
    case 1
        lm1 = ind1(1);
        hm1 = ind1(K(2));
        xmin(1) = xx1(sample) + round((lm1 + hm1) /2);

    case 2
        lm1 = ind1(1);
        hm2 = ind1(K(2));
        temp =0;
        for i = 2:K(2)
            if temp == 0 & abs( ind1(i) - ind1(i-1) ) > 1
                hm1 = ind1(i-1);
                lm2 = ind1(i);
                temp = +1;
            end;
        end;

        xmin(1) = xx1(sample) + round((lm1 + hm1) /2);
        xmin(2) = xx1(sample) + round((lm2 + hm2) /2);

    case 3
        lm3 = ind1(1);
        hm1 = ind1(K(2));
        temp =0;
        for i = 2:K(2)
            if temp == 0 & abs( ind1(i) - ind1(i-1) ) > 1
                hm3 = ind1(i-1);
                lm1 = ind1(i);
                temp = 1;
            end;
        end;

        xmin(1) = xx1(sample) + round((lm1 + hm1) /2)
        xmin(3) = xx1(sample) + round((lm3 + hm3) /2);
        otherwise
            disp('MINIMUMS FAILED');
end;

scatter(xmin(1),I2f(hsize/2+(n-
1)*hsize,xmin(1),color),100,colorr);hold all;
if xmin(2) ~= 0
    scatter(xmin(2),I2f(hsize/2+(n-
1)*hsize,xmin(2),color),100,'k');hold all;
end;
if xmin(3) ~= 0;
    scatter(xmin(3),I2f(hsize/2+(n-
1)*hsize,xmin(3),color),100,'k');hold all;
end;

```

```

temp1(n)=xmin(1)
temp2(n)=xmin(2)
temp3(n)=xmin(3)
    end;
xmin = [mean(temp1) mean(temp2) mean(temp3)]
%xmin(1) = mean(temp1)
%xmin(2) = mean(temp2)
%xmin(3) = mean(temp3)
if xmin(2)~= 0;
    rad_pix12(sample)= 2*pi/(xmin(2)-xmin(1))
end;
if xmin(3)~= 0
    rad_pix31(sample)= 2*pi/(xmin(1)-xmin(3))
end;

PX1(sample)=xmin(1)
PX2(sample)=xmin(2)
PX3(sample)=xmin(3)
end;
end;
if rerunning == 1 |rerunning == 2
    if rerunning == 2
        Nsample =Ntemp(1); % resetting Nsample
        sample = Nsample;
    end;
    ind2 = find(svolt == 100)
    ind12 = find(rad_pix12 ~= 0)
    %rad_pix12(ind12)= (PX2(ind12)-PX1(ind12))
    %rad_pix12(ind12) = 2*pi/rad_pix12(ind12)
    rad_12 =mean(rad_pix12(ind12))

    PX1_mean=mean(PX1(ind2))
    Phaseshift = mean(rad_pix12(ind12))*(PX1 - mean(PX1(ind2)))

    fig=fig+1; figure(fig);
    set(fig, 'Position', [1 scrsz(4)*0.5 scrsz(3)*.8 scrsz(4)*0.40]);
    set(fig, 'Name', ['Pixels of ', num2str(Nsample), 'files versus
Voltage', sfile{sample}, ' for ', scolorr{sample}]);
    plot(svolt, PX1, [scolorr{sample}, ':x']);hold all;
    plot(svolt, PX2, 'o');hold all;
    plot(svolt, PX3, '*');hold all;
    xlabel('Volts', 'FontSize', 16)
    ylabel('Pixel column', 'FontSize', 16)

    fig=fig+1; figure(fig);
    set(fig, 'Position', [1 scrsz(4)*0.03 scrsz(3)*.8 scrsz(4)*0.40]);
    set(fig, 'Name', ['Phase ', num2str(Phaseshift(sample)), ' of
', num2str(Nsample), ' versus Voltage', sfile{sample}, ' for
', scolorr{sample}]);
    plot(svolt, Phaseshift, [scolorr{sample}, ':*']);hold all;
    if exist('rad_pix31', 'var') == 1
        ind31 = find(rad_pix31 ~= 0)
        rad_31 =mean(rad_pix31(ind31))
        Phaseshift3 = mean(rad_pix31(ind31))*(PX1 -
mean(PX1(ind2)))

```

```
        plot(svolt,Phaseshift3,['k','x']);hold all;
end;
xlabel('Volts','FontSize',16)
ylabel('Phase [radians]','FontSize',16)
end;
toc
clear moon1
```

Appendix C: Run Sheet for EOAM v1.5b

Run Sheet for EOAM v1.5b

Third run of EOAM device wafers

Step 1)

Obtain 4" Silicon Wafers

Step 2)

RCA Clean of Silicon Wafers

Process Steps:

1) RCA Clean

- APM(SCI): 10 Minute Soak
- DI Water: 5 Minute Rinse
- Dilute HF: 1 Minute Soak
- DI Water: 5 Minute Rinse
- HPM: 10 Minute Soak
- DI Water: 5 Minute Rinse

2) Surface Defect Metrology Using the Tencor Surfscan tool.

Process Tools:

1) Wet Bench: RCA MOS Clean

2) Tencor 364 Surfscan

Step 3)

Growth of ~1 micron of Isolation Oxide

- Wet Oxide growth in Bruce Furnace (Tube 1)

Process Steps:

1) Thermal Processing in Bruce Furnace

- Run Program 888 (Furnace Warmup)

- Run Program 168 (1 micron wet oxide)

2) Oxide Thickness Metrology on Control Wafer C1

Process Tools:

1) Bruce Furnace (Tube 1)

2) Nanospec

Step 4)

Deposition of Metal #1

-Sputtering of ~0.5 μm Aluminum

Process Steps:

1) Aluminum Sputtering in CVC 601

-Argon Flow: 18.0 sccm

-Flow Pressure: 5.0 mTorr

-Power: 1500 Watts (approx. 460 V and 3.26 Amps)

-Target: 8" Target

2) Aluminum Thickness Metrology on Control Wafer

Sputtering Parameters:

1) Approximate Time: 1020 seconds

2) Desired Thickness: 0.5 μm

3) Resulting Thickness: 0.41 μm

Process Tools:

1) CVC 601 Sputter

2) Nanospec

Step 5)

Patterning of Mask #1 in Photoresist

-Suss MA 150 Aligner

Process Steps:

1) Coating of Photoresist on SVG Track

- Material: Shipley 1813

- Program: Program 1

- Thickness: 1.25 μm
- 2) Exposure in Suss MA 150 Aligner (broadband)
 - Mask Metal 1
 - Actual Dose: 76 mJ/cm^2
 - Resolution: 1.0 micron
- 3) Develop SVG Track
 - Program: 1
- 4) Photoresist Defect Metrology on Device Wafers

Process Tools:

- 1) SVG Track
- 2) Suss MA
- 3) Nanospec

Step 6)

Etching of Metal 1

Metal Wet Etch followed by ashing of Photoresist in Branson Asher

Process Steps:

- 1) Etch in Metal Wet Etch
 - When the tank has reached 50C, place the wafers in the tank until the aluminum clears and then give them an additional 10%.
 - After completing an aluminum etch, rinse the wafers for 5 minutes in the rinse tank before transferring to the rinsers/dryer.
- 2) Ash Wafers in Branson 3200 Asher
- 3) Pattern Defect Metrology

Etch Parameters:

- 1) Expected Time: 2.5 Minutes
- 2) Expected Rate: 2646 $\text{\AA}/\text{min}$

Process Tools:

- 1) Metal Wet Etch Bench
- 2) Branson 3200 Asher
- 3) Nanospec

Step 7)

Deposit Oxide Layer #2

CVD techniques via the AME P5000

Process Steps:

- 1) Deposit ~1 micron of Oxide
 - Process: 1 Micron TEOS Low Stress
 - Desired Thickness: 10064 Angstroms
- 2) Oxide Thickness Metrology on Dummy Wafer

Process Tools:

- 1) AME P5000
- 2) Nanometrics Spectrophotometers

Step 8)

Patterning of Mask #2 in photoresist

Suss MA 150 Aligner

Process Steps:

- 1) Coating of Photoresist on SVG Track
 - Material: Shipley 1813
 - Program: Program 1
 - Thickness: 1.2 μm
- 2) Exposure in Suss MA 150 Aligner (broadband)
 - Mask Via
 - Actual Dose: 103.2 mJ/cm²
 - Resolution: 1.0 micron
- 3) Develop SVG Track
 - Program: 1
- 4) Photoresist Defect Metrology on Device Wafers

Process Tools:

- 1) SVG Track
- 2) Suss MA

3) Nanospec

Step 9)

Etching of Via Pattern onto oxide Layer

BOE Wet Etch followed by ashing of photoresist in Branson 3200 Asher

Process Steps:

1-1) Etch in BOE

-Controlled Etch of Specified Time

-5 Minute Rinse

-Dry in Spin/Rinse Dryer

3) Pattern Defect Metrology

Etch Parameters:

1) Desired Rate: 586 A/min

2) Etch Rate: 1479 A/min

3) Etch Time: 7:40 min

Process Tools:

1) Drytek Quad

2) Branson 3200 Asher

3) Nanospec

Step 10)

Deposition of Metal #2

Sputtering of 1.50 - 2.00 μm Aluminum (**1.0 μm used in previous runs**)

Process Steps:

1) Aluminum Sputtering in CVC 601

-Argon Flow: 18.0 sccm

-Flow Pressure: 5.0 mTorr

-Power: 1500 Watts (approx. 460 V and 3.26 Amps)

-Target: 8" Target

2) Aluminum Thickness Metrology on Control Wafer

Sputtering Parameters:

1) Approximate Time: 3000 seconds **Adjust for thickness**

2) Desired Thickness: 1.50 μm

Process Tools:

1) CVC 601 Sputter

2) Nanospec

Step 11)

Patterning of Mask #2 in photoresist

Suss MA 150 Aligner

Process Steps:

1) Coating of Photoresist on SVG Track

- Material: Shipley 1813

- Program: Program 1

- Thickness: 1.2 μm

2) Exposure in Suss MA 150 Aligner (broadband)

-Mask Metal 2

-Actual Dose: 97.2 mJ/cm^2

-Resolution: 1.0 micron

3) Develop SVG Track

-Program: 1

4) Photoresist Defect Metrology on Device Wafers

Process Tools:

1) SVG Track

2) Suss MA

3) Nanospec

Step 12)

Etching of Metal 2

Metal Wet Etch followed by ashing of Photoresist in Branson Asher

Process Steps:

1) Etch in Metal Wet Etch

-When the tank has reached 50C, place the wafers in the tank until the aluminum clears and then give them an additional 10%.

-After completing an aluminum etch, rinse the wafers for 5 minutes in the rinse tank before transferring to the rinser/dryer.

2) Ash Wafers in Branson 3200 Asher

3) Pattern Defect Metrology

Etch Parameters:

1) Expected Time: 7.5 Minutes

2) Expected Rate: 2646 A/min

Process Tools:

1) Metal Wet Etch Bench

2) Branson 3200 Asher

3) Nanospec

Step 13)

Probe of EOAM devices

-Verify contact between pads and pixels

-Verify pads not shorted to each other

Step 14) (Newly developed process)

CMP of Metal 2

Objective:

Create mirror like finish of metal on pixel areas

Process Tools:

Strausbaugh

Step 15) (Newly developed process)

Objective:

Patterning of Spacer Layer in photoresist –12 microns thickness

Method:

Patterning of fourth photoresist mask using Suss MA 150 Aligner

Process Steps:

- 1) Coating of Photoresist on Hand Coater
 - Material: SU-8
 - Ramp up spinner to 2400 RPM for 5 s
 - Hold at 2400 RPM for 40 s
 - Ramp Down
- 2) Soft Bake at 95 Degrees Centigrade
 - Bake at 65 Degrees Centigrade for 2 minutes
 - Bake at 95 Degrees Centigrade for 5 minutes
- 3) Exposure in Suss Mask Aligner (365 nm)
 - Mask #4
 - Dose to Clear: approx. 180 mJ/cm²
 - Thickness: 20.0 micron
- 4) Post Exposure Bake
 - Bake at 65 Degrees Centigrade for 1 minute
 - Bake at 95 Degrees Centigrade for 2 minutes
- 5) Develop on wet bench using RER-600 Developer
 - Develop using RER-600 for 3 minutes
 - Rinse in IPO
- 6) Hard Bake
 - Bake at 65 Degrees Centigrade for 1 minute
 - Bake at 95 Degrees Centigrade for 2 minutes
 - Bake at 150 Degrees Centigrade for 5 minutes
- 6) Photoresist Defect Metrology on Device Wafers

Process Tools:

- 1) Hand Spinner
- 2) Wet Development Bench
- 3) Nanometrics Spectrophotometers

Step 16)

Objective:

Dice wafers into separate devices

Method: Dice wafer so that individual ‘good’ device have “handling die attached”.

Process Tools:

Tempress 4 Inch Wafer Saw

Step 17) (Newly developed process)

Objective:

Fill Layer with PDLC and cover with ITO Glass Slide

Method:

The cell will be filled with PDLC manually before a prefabricated ITO glass slide is placed over the lens area.

Step 18) (Newly developed process)

Objective:

UV Exposure to cure

Method:

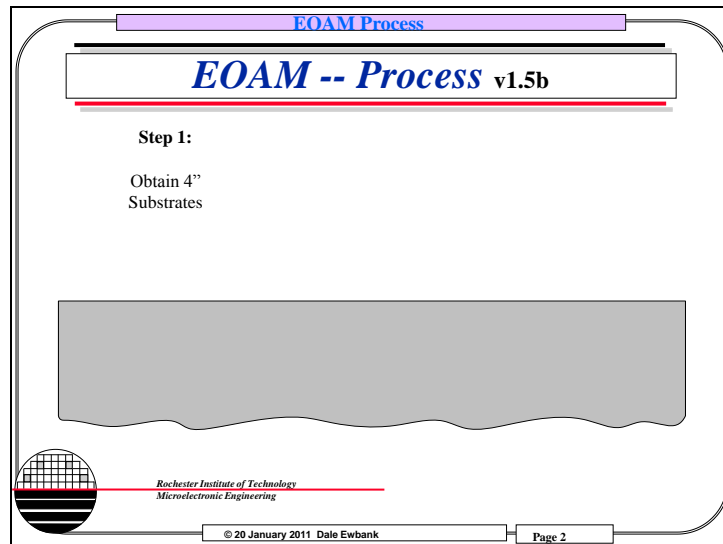
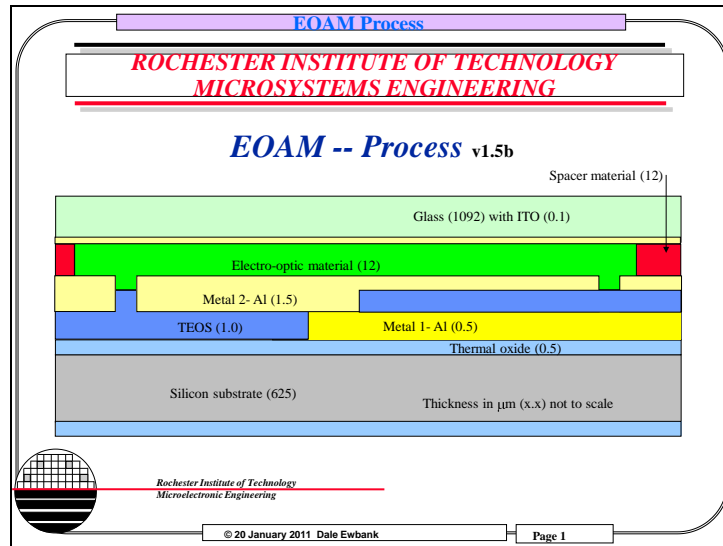
Completed dvice will be exposed to ultraviolet light via exposure.

Procedure:

1.) UV Exposure with i-line head (**Newly developed tool**)

- Required Dose: **minimum 50 mW/cm² for 120 seconds**


Appendix D: EOAM_Process Rev1_5b




EOAM Process

***EOAM -- Process* v1.5b**

Step 2:
RCA Clean



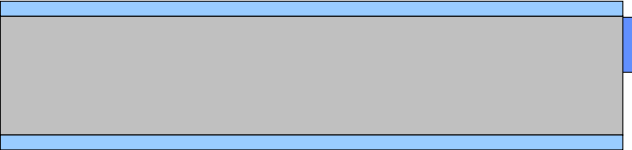
 *Rochester Institute of Technology*
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 3

EOAM Process

***EOAM -- Process* v1.5b**

Step 3:
Grow >500 nm Thermal Oxide




 *Rochester Institute of Technology*
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 4

EOAM Process

***EOAM -- Process* v1.5b**

Step 4:
Deposit Metal 1




 *Rochester Institute of Technology*
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 5

EOAM Process

***EOAM -- Process* v1.5b**

Step 5:
Pattern photoresist for Metal 1



 *Rochester Institute of Technology*
Microelectronic Engineering

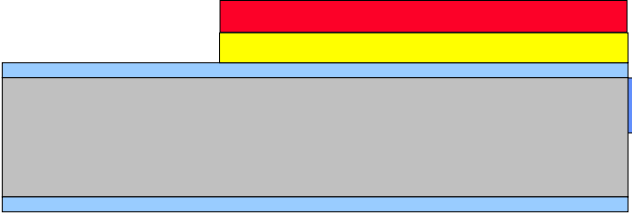
© 20 January 2011 Dale Ewbank Page 6


EOAM Process

EOAM -- Process v1.5b

Step 6:

Etch Metal 1



 Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank


Page 7


EOAM Process

EOAM -- Process v1.5b

Step 6 continued :

Strip Resist



 Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank

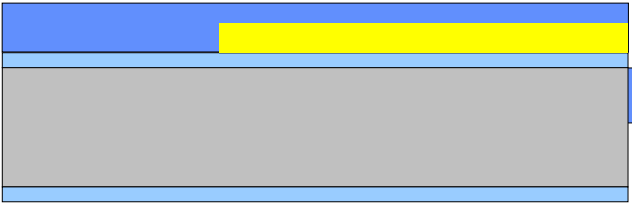
Page 8


EOAM Process

EOAM -- Process v1.5b

Step 7:

Deposit PECVD of TEOS Silicon Dioxide



 Rochester Institute of Technology
Microelectronic Engineering

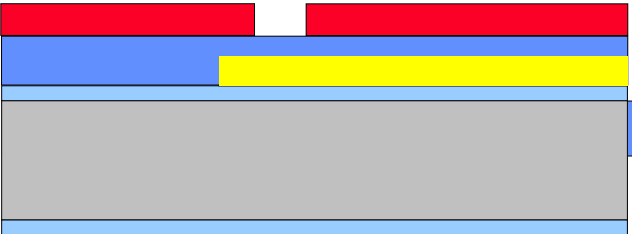
© 20 January 2011 Dale Ewbank Page 9


EOAM Process

EOAM -- Process v1.5b

Step 8:

Pattern photoresist for Vias



 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 10

EOAM Process

EOAM -- Process v1.5b

Step 9:

Etch Vias into TEOS

 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 11

EOAM Process

EOAM -- Process v1.5b

Step 9 continued:

Strip resist

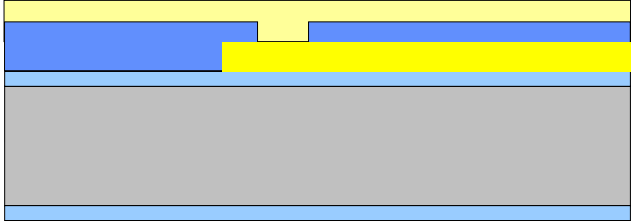
 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 12

EOAM Process

EOAM -- Process v1.5b

Step 10:
Deposit Metal 2



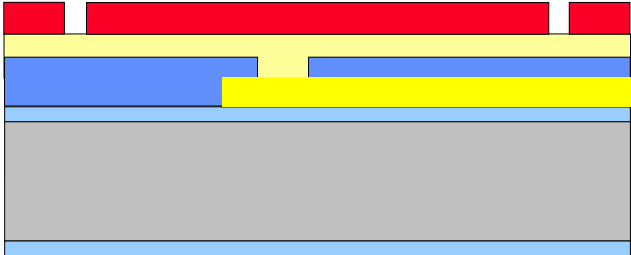
 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 13

EOAM Process

EOAM -- Process v1.5b

Step 11:
Pattern Photoresist for Metal 2




 Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank Page 14

EOAM Process

EOAM -- Process v1.5b

Step 12:
Etch Metal 2


 Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank Page 15

EOAM Process

EOAM -- Process v1.5b

Step 12 continued :
Strip resist

 Rochester Institute of Technology
Microelectronic Engineering

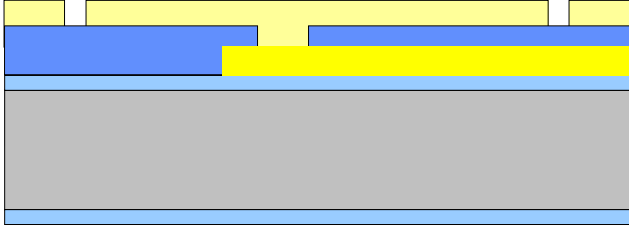
© 20 January 2011 Dale Ewbank Page 16

EOAM Process


***EOAM -- Process* v1.5b**

Step 13:

Probe to verify contact between pads and pixels



The diagram shows a cross-section of a microchip. At the top, there are yellow rectangular pads on a blue layer. A yellow probe tip is shown touching one of these pads. Below the blue layer is a light blue layer, and at the bottom is a thick grey substrate. A thin blue layer is visible at the very bottom of the substrate.

 Rochester Institute of Technology
Microelectronic Engineering

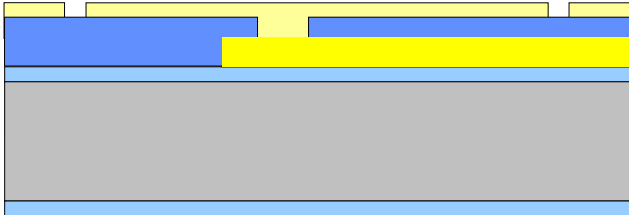
© 20 January 2011 Dale Ewbank Page 17

EOAM Process


***EOAM -- Process* v1.5b**

Step 14:

CMP Metal 2 (Newly developed process)



The diagram shows a cross-section of a microchip, similar to the previous one. It features yellow rectangular pads on a blue layer. The probe tip is no longer present. The layers below (light blue and grey substrate) are the same as in the previous step.

 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 18

EOAM Process

***EOAM -- Process* v1.5b**

Step 15:

Pattern Spacer layer (Newly developed process)

 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 19

EOAM Process

***EOAM -- Process* v1.5b**

Step 16:

Dice wafer into separate devices

 Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 20

EOAM Process

***EOAM -- Process* v1.5b**

Step 17:

Fill with PDLC precursor (**Newly developed process**)


Rochester Institute of Technology
Microelectronic Engineering


© 20 January 2011 Dale Ewbank Page 21

EOAM Process

***EOAM -- Process* v1.5b**

Step 17 continued:

Cover with ITO glass slide (**Newly developed process**)


Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank Page 22

EOAM Process

EOAM -- Process v1.5b

Step 18:

UV Exposure to cure PDLC (Newly developed process and tool)

Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank Page 23

EOAM Process

EOAM layers

EOAM -- Process v1.5b

Rochester Institute of Technology
Microelectronic Engineering

© 20 January 2011 Dale Ewbank Page 24

Appendix E: g5_nm_PDLC.fig and g5_nm_PDLC.m

```
function varargout = g5_nm_PDLC(varargin)
%***** updated 01082009 DEE see line 409
% to correct complex number display
% G5_NM_PDLC M-file for g5_nm_PDLC.fig
% G5_NM_PDLC, by itself, creates a new G5_NM_PDLC or raises the
existing
% singleton*.
%
% H = G5_NM_PDLC returns the handle to a new G5_NM_PDLC or the
handle to
% the existing singleton*.
%
% G5_NM_PDLC('CALLBACK',hObject,eventData,handles,...) calls the
local
% function named CALLBACK in G5_NM_PDLC.M with the given input
arguments.
%
% G5_NM_PDLC('Property','Value',...) creates a new G5_NM_PDLC or
raises
% the existing singleton*. Starting from the left, property value
pairs are
% applied to the GUI before g4_nm_Dvary_OpeningFunction gets
called. An
% unrecognized property name or invalid value makes property
application
% stop. All inputs are passed to g5_nm_PDLC_OpeningFcn via
varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help g5_nm_PDLC

% Last Modified by GUIDE v2.5 29-Sep-2010 14:30:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @g5_nm_PDLC_OpeningFcn, ...
                  'gui_OutputFcn',  @g5_nm_PDLC_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```

else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before g5_nm_PDLC is made visible.
function g5_nm_PDLC_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to g5_nm_PDLC (see VARARGIN)

% Choose default command line output for g5_nm_PDLC
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

initialize_gui(hObject, handles, false);

% UIWAIT makes g5_nm_PDLC wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = g5_nm_PDLC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% function selectlayer_SelectionChangeFcn(hObject, eventdata, handles)
% % hObject    handle to unitgroup (see GCBO)
% % eventdata  reserved - to be defined in a future version of MATLAB
% % handles    structure with handles and user data (see GUIDATA)
% set(handles.text4, 'String', 'setting');
% switch hObject
%     case handles.layer3
%         handles.metricdata.layer=3;
%     case handles.layer4
%         handles.metricdata.layer=4;
%     case handles.layer5
%         handles.metricdata.layer=5;
%     otherwise
%         handles.metricdata.layer=2;    %handles.layer2
% end
% guidata(hObject,handles)
% set(handles.R, 'String',handles.metricdata.layer);
% guidata(handles.figure1, handles);

% -----

```

```

% --- Executes during object creation, after setting all properties.
function n1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

function n1_Callback(hObject, eventdata, handles)
% hObject    handle to n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n1 as text
%       str2double(get(hObject,'String')) returns contents of n1 as a
double
n1 = str2double(get(hObject, 'String'));
if isnan(n1)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.n1 = n1;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

function m1_Callback(hObject, eventdata, handles)

```

```

% hObject    handle to m1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m1 as text
%         str2double(get(hObject,'String')) returns contents of m1 as a
double
m1 = (get(hObject, 'String'));
if ischar(m1)
    set(hObject, 'String', 0);
    errordlg('Input must be a char','Error');
end

% Save the new m1 value
handles.metricdata.m1 = m1;
guidata(hObject,handles)

% --- Executes on button press in calculate.
function calculate_Callback(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axes(handles.axes1);
cla;
%
%function multilayer_Guenther_interface_7resist
%DEE 02/06/2007
%     Calculates Reflectance from film stack at angle of incidence.
%     Based on Guenther "Modern Optics" pg. 120-128
%
% set-up parameters
%global E1 E2 media_name n stack D lambda k r t tD
Aname='multilayer_Guenther_interface_7resist.m';
media_name=[handles.metricdata.m1 handles.metricdata.m2
handles.metricdata.m3 ...
            handles.metricdata.m4 handles.metricdata.m5
handles.metricdata.m6]
%
% Input for multilayer film stack
% n >> complex index of refraction
% D >> thickness in [nm]
% lambda >> units of [nm]
% *****
n=[ handles.metricdata.n1 handles.metricdata.n2 handles.metricdata.n3
...
    handles.metricdata.n4 handles.metricdata.n5 handles.metricdata.n6
]
D=[ handles.metricdata.z1 handles.metricdata.z2 handles.metricdata.z3
...
    handles.metricdata.z4 handles.metricdata.z5 handles.metricdata.z6]
Drange=400
lambda=handles.metricdata.lambda
theta_id=handles.metricdata.angleI
% theta_id=10 %degrees
Dvary=handles.metricdata.layer    % choose layer 2 through inter-1

```

```

interface=handles.metricdata.Ninterfaces      % choose interface for
reflectance calculation of stack

stack=size(n)
Rstack=zeros(2,stack(2));
Dtemp=D(Dvary)
theta_i=theta_id*pi/180    % [degrees]*pi/180 = [radians]
                        % [radians] angle of incidence in layer 1 (top
layer of stack "usually air")
EfTE=[1
      0]    % E field final TE
EfTM=[1
      0]    % E field final TM
% set angle of illumination *****
theta=ones(1,stack(2))*99;
theta(1)=theta_i;
for j=2 : stack(2)          % what happens if theta >= 90
degrees?
    theta(j)=asin((n(j-1)/n(j))*sin(theta(j-1)));
end;

% for inter = 2 : interface
inter = interface
    if Dtemp > 500
        Dmin=Dtemp-Drange/2;
        Dmax=Dtemp+Drange/2;
    else
        Dmin=1;
        Dmax=Dtemp+1;
    end;
    Erange=Dmax-Dmin+1
    Evary=zeros(2,1,Erange);

r=zeros(1,inter);          t=ones(1,inter); % fresnel coefficients
rho=ones(1,inter);        % reflection coefficient
for stack

Es=ones(2,1,inter)*7;     Ep=ones(2,1,inter)*6;    % E-field at bottom
of layer
I=ones(2,2,inter)*9;      % interface transfer matrix
T=ones(2,2,inter)*99;     % bulk film transfer matrix (phase and
absorption??)
M=zeros(2,2,inter);       % composite I*T for each transfer layer

Mfs=[1 0
     0 1];
Mfp=[1 0
     0 1];

    % run for TE pol=1 and then for TM pol=2 polarization
for pol = 1:2
    if pol ==1
        N=n.*cos(theta)    % complex effective index TE for each
layer
    else
        N=n./cos(theta)    % complex effective index TM for each
layer

```

```

        end;
delta=2.*pi.*n.*cos(theta).*D/lambda    % phase difference for single
pass thru layer
for vary= Dmin : Dmax
D(Dvary)=vary-1;
%delta(Dvary)=2*pi*N(Dvary)*D(Dvary)/lambda;    % phase for effective
N?????
delta(Dvary)=2*pi*n(Dvary)*cos(theta(Dvary))*D(Dvary)/lambda;    % phase
difference for single pass thru layer
%
    for j=2:inter
        r(j)=(N(j-1)-N(j))/(N(j-1)+N(j));
        t(j)= 2*N(j-1)/(N(j-1)+N(j));

        I(:, :, j)=[1/t(j)  r(j)/t(j)
                    r(j)/t(j)  1/t(j)];
        T(:, :, j)=[exp(i*delta(j))  0
                    0  exp(-i*delta(j))];

        M(:, :, j)=I(:, :, j)*T(:, :, j);

    end;
MT=[1  0
     0  1];
    for j= 2:inter
        if j == inter
            MT=MT*I(:, :, j);
        else
            MT=MT*I(:, :, j)*T(:, :, j) ;
        end;
    end;
    if pol == 1
        Es=MT*EfTE;
        Evary(:, :, (vary-Dmin+1))=Es;
        rhoTE=Es(2,1)./Es(1,1);
        RstackTE=rhoTE.*conj(rhoTE);
        RTE(vary-Dmin+1)=RstackTE;
        if (vary-Dmin+1) == (Erange-1)/2
            Rstack(1,inter)=RstackTE;
        end;
    else
        Ep=MT*EfTM;
        Evary(:, :, (vary-Dmin+1))=Ep;
        rhoTM=Ep(2,1)./Ep(1,1);
        RstackTM=rhoTM.*conj(rhoTM);
        RTM(vary-Dmin+1)=RstackTM;
        if (vary-Dmin+1) == (Erange-1)/2
            Rstack(2,inter)=RstackTM;
        end;
    end;
end;

if pol ==1

%         figure(inter)
%         set(inter,'Name',['Display from ',Aname]);

```



```

        plot((Dmin:1:(Dmax)),RTE,'b'); hold all;
        title(['Reflectance of ',num2str(inter),' layer film stack vs
layer ',num2str(Dvary),' thickness']); hold all;
        [maxRTE,ImaxRTE]=max(RTE);
        [minRTE,IminRTE]=min(RTE);
        Fresnel_rTE=r
        ReflectanceTE=r.*conj(r)
    else
        [maxRTM,ImaxRTM]=max(RTM);
        [minRTM,IminRTM]=min(RTM);
    %
        figure(inter)
        plot((Dmin:1:(Dmax)),RTM,'-.g'); hold all;
        xlabel(['layer ',num2str(Dvary),' thickness
[nm]'],'FontSize',10);
        ylabel(['Reflectance (incidence of ',num2str(theta_id),'
degrees)'],'FontSize',10);
        legend(['TE ',num2str(minRTE),' at ',num2str(IminRTE-
1+Dmin),...
                'nm ',num2str(maxRTE),' at ',num2str(ImaxRTE-
1+Dmin),'nm ',num2str(RTE(Erange))],...
                ['TM ',num2str(minRTM),' at ',num2str(IminRTM-
1+Dmin),...
                'nm ',num2str(maxRTM),' at ',num2str(ImaxRTM-
1+Dmin),'nm ',num2str(RTM(Erange))]);
        legend('location','Best');
        legend('boxoff');
        Fresnel_rTM=r
        ReflectanceTM=r.*conj(r)
    end;
end;
D(Dvary)=Dtemp;

media_name
n
% end;
ReflectanceTE
ReflectanceTM
R=[ReflectanceTE ReflectanceTM]
set(handles.R, 'String', Dvary);

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

initialize_gui(gcf, handles, true);

% -----

function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it
means

```

```

% we are we are just re-initializing a GUI by calling it from the cmd
line
% while it is up. So, bail out as we dont want to reset the data.
if isfield(handles, 'metricdata') && ~isreset
    return;
end
handles.metricdata.m1 = 'air';
handles.metricdata.m2 = 'glass';
handles.metricdata.m3 = 'ITO';
handles.metricdata.m4 = 'PDL';
handles.metricdata.m5 = 'Al';
handles.metricdata.m6 = 'SiO2';
handles.metricdata.Ninterfaces = 6;
handles.metricdata.n1 = 1;
handles.metricdata.n2 = 1.5;
handles.metricdata.n3 = 1.93-0.0i;
handles.metricdata.n4 = 1.6-0.0i;
handles.metricdata.n5 = 0.769-6.08i;
handles.metricdata.n6 = 4.297488-.07i;
handles.metricdata.z1 = 1000;
handles.metricdata.z2 = 100000;
handles.metricdata.z3 = 120;
handles.metricdata.z4 = 10000;
handles.metricdata.z5 = 500;
handles.metricdata.z6 = 10000;
handles.metricdata.lambda = 541;
handles.metricdata.angleI = 2;
handles.metricdata.layer = 3;
plot(handles.metricdata.n1,handles.metricdata.z1);

set(handles.m1, 'String', handles.metricdata.m1);
set(handles.m2, 'String', handles.metricdata.m2);
set(handles.m3, 'String', handles.metricdata.m3);
set(handles.m4, 'String', handles.metricdata.m4);
set(handles.m5, 'String', handles.metricdata.m5);
set(handles.m6, 'String', handles.metricdata.m6);
set(handles.Ninterfaces, 'String', handles.metricdata.Ninterfaces);
set(handles.n1, 'String', num2str(handles.metricdata.n1));
set(handles.n2, 'String', num2str(handles.metricdata.n2));
set(handles.n3, 'String', num2str(handles.metricdata.n3));
set(handles.n4, 'String', num2str(handles.metricdata.n4));
set(handles.n5, 'String', num2str(handles.metricdata.n5));
set(handles.n6, 'String', num2str(handles.metricdata.n6));
set(handles.z1, 'String', handles.metricdata.z1);
set(handles.z2, 'String', handles.metricdata.z2);
set(handles.z3, 'String', handles.metricdata.z3);
set(handles.z4, 'String', handles.metricdata.z4);
set(handles.z5, 'String', handles.metricdata.z5);
set(handles.z6, 'String', handles.metricdata.z6);
set(handles.lambda, 'String', handles.metricdata.lambda);
set(handles.angleI, 'String', handles.metricdata.angleI);
set(handles.R, 'String', 0);

set(handles.selectlayer, 'SelectedObject', handles.layer3);

set(handles.text4, 'String', '[nm] <g5_nm_PDL> 2010 DEE');

```

```

% Update handles structure
guidata(handles.figure1, handles);

function z1_Callback(hObject, eventdata, handles)
% hObject    handle to z1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z1 as text
z1=      str2double(get(hObject,'String')) %returns contents of z1 as
a double
if isnan(z1)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
% Save the new z1 value
handles.metricdata.z1 = z1;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function z1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

function lambda_Callback(hObject, eventdata, handles)
% hObject    handle to lambda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of lambda as text
%         str2double(get(hObject,'String')) returns contents of lambda
as a double
lambda=         str2double(get(hObject,'String')) %returns contents of
lambda as a double
if isnan(lambda)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
% Save the new lambda value
handles.metricdata.lambda = lambda;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function lambda_CreateFcn(hObject, eventdata, handles)
% hObject     handle to lambda (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
% if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
%     set(hObject,'BackgroundColor','white');
% end
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% new variablessss      *****

function m2_Callback(hObject, eventdata, handles)
% hObject     handle to m2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m2 as text
%         str2double(get(hObject,'String')) returns contents of m2 as a
double
m2 = (get(hObject, 'String'));
if ischar(m2)
    set(hObject, 'String', 0);
    errordlg('Input must be a char','Error');
end

% Save the new m2 value
handles.metricdata.m2 = m2;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m2_CreateFcn(hObject, eventdata, handles)
% hObject     handle to m2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB

```

```

% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m3_Callback(hObject, eventdata, handles)
% hObject      handle to m3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m3 as text
%      str2double(get(hObject,'String')) returns contents of m3 as a
double
m3 = (get(hObject, 'String'));
if ischar(m3)
    set(hObject, 'String', 0);
    errordlg('Input must be a char','Error');
end

% Save the new m3 value
handles.metricdata.m3 = m3;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m3_CreateFcn(hObject, eventdata, handles)
% hObject      handle to m3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m4_Callback(hObject, eventdata, handles)
% hObject      handle to m4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m4 as text
%      str2double(get(hObject,'String')) returns contents of m4 as a
double
m4 = (get(hObject, 'String'));
if ischar(m4)
    set(hObject, 'String', 0);

```

```

        errordlg('Input must be a char','Error');
end

% Save the new m4 value
handles.metricdata.m4 = m4;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function m5_Callback(hObject, eventdata, handles)
% hObject    handle to m5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m5 as text
%         str2double(get(hObject,'String')) returns contents of m5 as a
double
m5 = (get(hObject, 'String'));
if ischar(m5)
    set(hObject, 'String', 0);
    errordlg('Input must be a char','Error');
end

% Save the new m5 value
handles.metricdata.m5 = m5;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function m6_Callback(hObject, eventdata, handles)
% hObject    handle to m6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m6 as text
%        str2double(get(hObject,'String')) returns contents of m6 as a
double
m6 = (get(hObject, 'String'));
if ischar(m6)
    set(hObject, 'String', 0);
    errordlg('Input must be a char','Error');
end

% Save the new m6 value
handles.metricdata.m6 = m6;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function m6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function n2_Callback(hObject, eventdata, handles)
% hObject    handle to n2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n2 as text
%        str2double(get(hObject,'String')) returns contents of n2 as a
double
n2 = str2double(get(hObject, 'String'));
if isnan(n2)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end

handles.metricdata.n2 = n2;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function n2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function n3_Callback(hObject, eventdata, handles)
% hObject      handle to n3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n3 as text
%      str2double(get(hObject,'String')) returns contents of n3 as a
double
n3 = str2double(get(hObject, 'String'));
if isnan(n3)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.n3 = n3;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function n3_CreateFcn(hObject, eventdata, handles)
% hObject      handle to n3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function n4_Callback(hObject, eventdata, handles)
% hObject      handle to n4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n4 as text
%      str2double(get(hObject,'String')) returns contents of n4 as a
double
n4 = str2double(get(hObject, 'String'));
if isnan(n4)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end

```



```

handles.metricdata.n4 = n4;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function n4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function n5_Callback(hObject, eventdata, handles)
% hObject    handle to n5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n5 as text
%         str2double(get(hObject,'String')) returns contents of n5 as a
double
n5 = str2double(get(hObject, 'String'));
if isnan(n5)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.n5 = n5;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function n5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function n6_Callback(hObject, eventdata, handles)
% hObject    handle to n6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n6 as text

```

```

%         str2double(get(hObject,'String')) returns contents of n6 as a
double
n6 = str2double(get(hObject, 'String'));
if isnan(n6)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.n6 = n6;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function n6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z2_Callback(hObject, eventdata, handles)
% hObject    handle to z2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z2 as text
%         str2double(get(hObject,'String')) returns contents of z2 as a
double
z2 = str2double(get(hObject, 'String'));
if isnan(z2)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.z2 = z2;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function z2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function z3_Callback(hObject, eventdata, handles)
% hObject    handle to z3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z3 as text
%        str2double(get(hObject,'String')) returns contents of z3 as a
double
z3 = str2double(get(hObject, 'String'));
if isnan(z3)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.z3 = z3;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function z3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z4_Callback(hObject, eventdata, handles)
% hObject    handle to z4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z4 as text
%        str2double(get(hObject,'String')) returns contents of z4 as a
double
z4 = str2double(get(hObject, 'String'));
if isnan(z4)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.z4 = z4;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function z4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.

```

```

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z5_Callback(hObject, eventdata, handles)
% hObject     handle to z5 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z5 as text
%         str2double(get(hObject,'String')) returns contents of z5 as a
double
z5 = str2double(get(hObject, 'String'));
if isnan(z5)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.z5 = z5;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function z5_CreateFcn(hObject, eventdata, handles)
% hObject     handle to z5 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function z6_Callback(hObject, eventdata, handles)
% hObject     handle to z6 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z6 as text
%         str2double(get(hObject,'String')) returns contents of z6 as a
double
z6 = str2double(get(hObject, 'String'));
if isnan(z6)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.metricdata.z6 = z6;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.

```

```

function z6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % --- Executes on button press in radiobutton20.
% function radiobutton20_Callback(hObject, eventdata, handles)
% % hObject    handle to radiobutton20 (see GCBO)
% % eventdata  reserved - to be defined in a future version of MATLAB
% % handles    structure with handles and user data (see GUIDATA)
%
% % Hint: get(hObject,'Value') returns toggle state of radiobutton20

function angleI_Callback(hObject, eventdata, handles)
% hObject    handle to angleI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
angleI=      str2double(get(hObject,'String')) %returns contents of
angleI as a double
if isnan(angleI)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
% Save the new angleI value
handles.metricdata.angleI = angleI;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function angleI_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angleI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% -----
function selectlayer_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to selectlayer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text4, 'String', 'setting');
switch hObject
    case handles.layer3
        handles.metricdata.layer=3;
    case handles.layer4
        handles.metricdata.layer=4;
    case handles.layer5
        handles.metricdata.layer=5;
    otherwise
        handles.metricdata.layer=2;    %handles.layer2
end
guidata(hObject,handles)

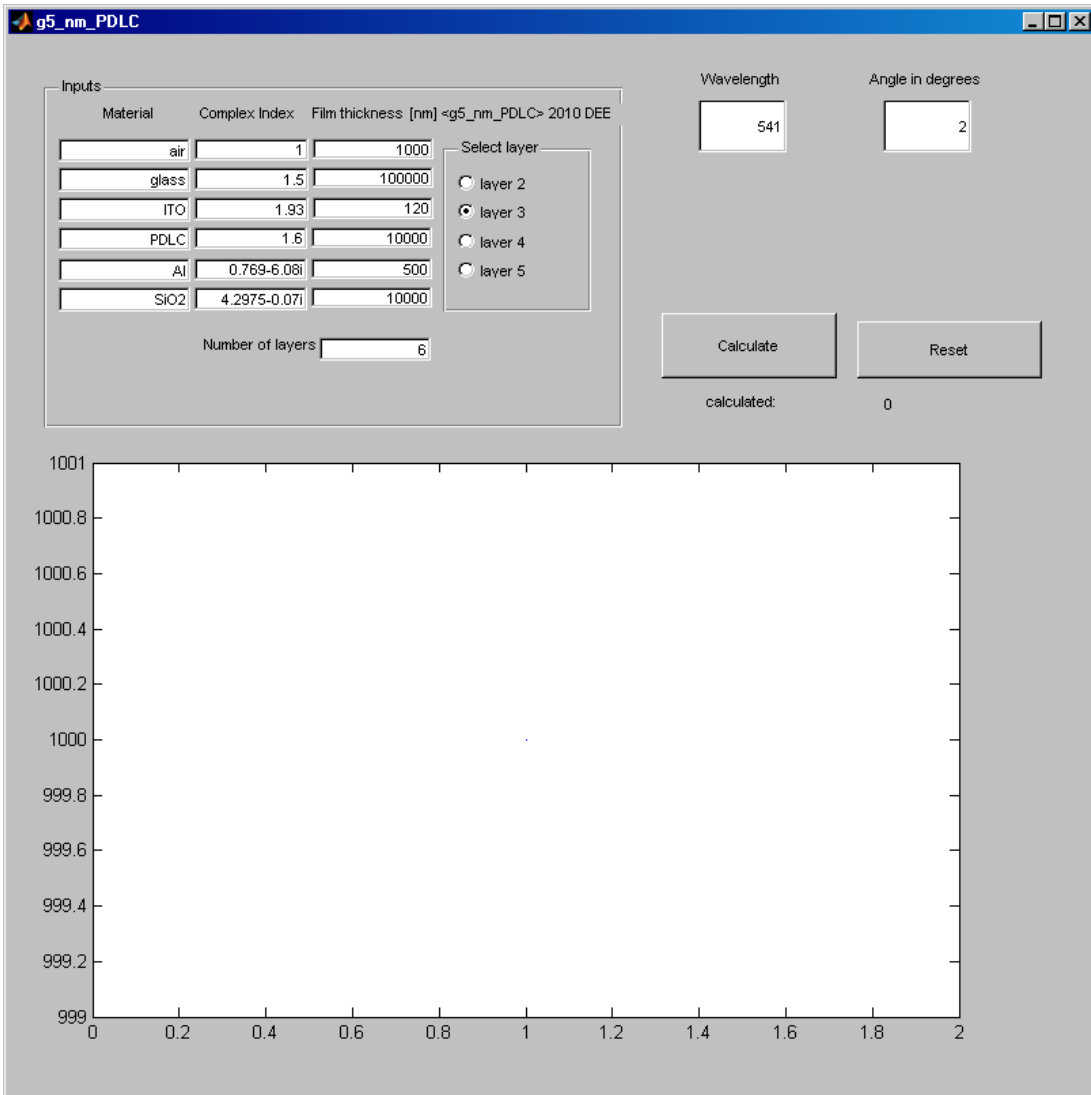
function Ninterfaces_Callback(hObject, eventdata, handles)
% hObject    handle to Ninterfaces (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ninterfaces as text
%        str2double(get(hObject,'String')) returns contents of
Ninterfaces as a double
Ninterfaces=        str2double(get(hObject,'String')) %returns contents
of Ninterfaces as a double
if isnan(Ninterfaces)
    set(hObject, 'String', 5);
    errordlg('Input must be a number','Error');
end
% Save the new Ninterfaces value
handles.metricdata.Ninterfaces = Ninterfaces;
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function Ninterfaces_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ninterfaces (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```



Appendix F: xu_2007_ph_ext_05182010_data_p.m

```
%function xu_2007_ph_ext_05182010_data_p.m
%Dale Ewbank from xu_2007.pdf
% case 'g'
%     wave = 0.5435;
%     color = 2;colorr='g';

% select files and assign voltage and color for Phase vs Voltage
analysis
% for jpg from Nikon D50 of single arm interferometer
% NOTE: For Ao > Ar see line 204 *****
%
close all hidden;
imtool close all;
global Nsample fig dataout
tic
clear IR C1 C2 dataout datahold Iout
set(0, 'Units', 'pixels')
scrsz = get(0, 'ScreenSize')
testing=0;

Iaverage=3;
sam=1;
xxbegin=800;
xxstep=500;
xxtimes=1;
xxend=xxbegin+(xxtimes-1)*xxstep;
yybegin=300;
yystep=xxstep;
yytimes=1;
yyend=yybegin+(yytimes-1)*yystep;
step=0;
for xx = xxbegin:xxstep:xxend
    for yy = yybegin:yystep:yyend
        % Pixel to select
        step=step+1;
            xxx=0 + xx; %;
            yyy=yy; %
            blocksize=512;
            dALPHA= 0.00001;    %change in calculated ALPHA to stop
iteration
            compare=1 ; % number of file to compare to
            Aname='xu_2007_ph_ext_05182010_data_p'
            %see kend to stop loop ~line 177
            pixel=[xxx yyy blocksize blocksize];
            I=zeros(blocksize,blocksize,2);
            Istdev=zeros(blocksize,blocksize,2);

rerunning = 0; % 0 for rerun with data in workspace, 2-98 for new
```



```

if xx == xxbegin & yy==yybegin

prompt = {'Running data files(0 for start iterations, 2-98, 99 for
Itest):',...
        'number of file to compare to: '}
mtitle = ['Input ',Aname];
lines= 1;
def      = {num2str(rerunning),num2str(compare)};
datas = inputdlg(prompt,mtitle,lines,def);
if isempty(datas)
    close all;
    disp(['User canceled ',Aname])
    return;
end;
rerunning =str2num(datas{1});
compare=str2num(datas{2});
end; % xx if
%         if rerunning == 2
%             prompt = {'ReRunning file2#: '}
%             mtitle = ['Input ',Aname];
%             lines= 1;
%             def      = {'99'};
%             datas = inputdlg(prompt,mtitle,lines,def);
%             if isempty(datas)
%                 close all;
%                 disp(['User canceled ',Aname])
%                 return;
%             end;
%             Nsingle =str2num(datas{1});
%         end;
if rerunning ==99 % 99 is for test algorithm
    scolorr='r';
    testing=99;
    svolt=[0 100];
    xx=xxend;
end;
if rerunning >= 0
    Aname='xu_2007_ph_ext_05182010_data_p';
    filestouse='';
    hsize = 8;
    fig =1;
    fig_reuseask=0;
    fig_reuse2ask=1;
    %     sam=1;
    Nsample = 98;

    Iaverage=3;
    %     scolorr='r';
    %     % Pixel to select
    %     xxx=1200;
    %     yyy=870;
    %     blocksize=32;
    %     pixel=[xxx yyy blocksize blocksize]
    %     I=zeros(blocksize,blocksize,2);
    %     Istdev=zeros(blocksize,blocksize,2);
end;

```

```

while rerunning >= 1 & rerunning <= 98
    svoltt=0;
    scolorr='g'; % green
    while sam <= Nsample & Iaverage ~= 99
        prompt = {'Averaging data files(1 for start, 2 or more), 99
for done:',...
'Enter Voltage: ', 'Enter color> b g or r: '}
        mtitle = ['Input ',Aname];
        lines= 1;
        def = {num2str(Iaverage), num2str(svoltt), scolorr};
        datas = inputdlg(prompt,mtitle,lines,def);
        if isempty(datas)
            disp(['User canceled inputs for',Aname])
            datas{1}='99';datas{2}='0';datas{3}=scolorr;
            sfiles='endoffiles'
%             return;
            end;
            Iaverage =str2num(datas{1});
            svoltt=str2num(datas{2});
            scolorr=datas{3};
            if Iaverage ~= 99
                [Iarray, I(:, :, sam), Istdev(:, :, sam), svolt(sam), scolorr,
sfiles]=singlelearn_Iaverage_10282008_noplot...
                (svoltt, Iaverage, scolorr, pixel, filestouse);
                end;
                Afiles(1,sam) = {sfiles};
            if sam > 1 & Iaverage ~= 99
                svoltt= svolt(sam)+(svolt(sam)-svolt(sam-1));
            end;
            if Iaverage ==99
                rerunning=0;
                Nsample=sam-1;
            end;
            sam=sam+1;
        end;
%         rerunning=1;
    end;
switch scolorr
    case 'b'
        wave = 0.4861327;
        color = 3;colorr='b';
    case 'g'
        wave = 0.5435;
        color = 2;colorr='g';
    case 'r'
        wave = 0.6562816;
        color = 1;colorr='r';
    otherwise
        disp(' Error on color input, please restart')
        return;
    end;
if testing ==99

```

```

M=2
else
    Ntemp=size(Afiles);
    Nsample =Ntemp(2)-1;
    filestouse=Afiles;
    M=Nsample    ;

    [Iarray, I(:, :,1), Istdev(:, :,1), svolt(compare), scolorr,
sfiles]=singlearm_Iaverage_10282008_noplot...
        (svolt(compare), Iaverage, scolorr, pixel,
filestouse{compare});
end;
dataout=zeros(M,10);
for jjj= 1:M
if jjj ~= compare
if rerunning ~= 99
    [Iarray, I(:, :,2), Istdev(:, :,2), svolt(jjj), scolorr,
sfiles]=singlearm_Iaverage_10282008_noplot...
        (svolt(jjj), 3, scolorr, pixel, filestouse{jjj});
    [X,Y] = meshgrid(1:1:blocksize);
figure(1);
    surf(X,Y,I(:, :,1)); %hold all;
    xlabel('I(:, :,1) versus X,Y ');
% end;
    figure(2);
    surf(X,Y,I(:, :,2)); %hold all;
    xlabel('I(:, :,2) versus X,Y ');
end;
% % if jjj==2
% %     rerunning == 1 % |rerunning == 2
% %     clear dd1 dd2 temp D DD C CC S SS
% % %     Ntemp=size(I);
% % %     Nsample =Ntemp(3)
% % temp_svolt=svolt
%     if testing ~=4
%         Ntemp=size(Afiles);
%         Nsample =Ntemp(2)-1;
%         filestouse=Afiles;
%     end;
%     [Iarray, I(:, :,1), Istdev(:, :,1), svolt(1), scolorr,
sfiles]=singlearm_Iaverage_0802...
%         (svolt(compare), 3, scolorr, pixel, filestouse{compare});

% if rerunning ==1
%     clear fit12w fit13w fit17w plotI plotIstdev
%     djCopy=[ 0:pi/M:(M-1)*pi/M]
%     djCopy=djCopy'
%     dj=djCopy
% end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start here when analyzing data
if rerunning == 1 | 99 % 99 is for test algorithm

```

```

% data for test input*****

[X,Y] = meshgrid(1:1:blocksize);
xmax=blocksize;
ymax=blocksize;
Ao=8 ;%*rand ; % Ao=1*(X+Y)/(2*blocksize)
Ar=ones (blocksize,blocksize,1)*3.0;

test_phi=pi*.3; %test_phi=pi*rand ; %radians
test_alpha= pi*.2 ; %radians
if rerunning == 99 & jjj==2
    arb=pi/4; %arbitrary initial phase
    I=zeros (blocksize,blocksize,2);
    rXY=rand(blocksize,blocksize)*0.05; noise=10;
    test_phi=0.6*((2*pi*X/xmax).^2+5*(2*pi*Y/ymax))+rXY ; %radians
    test_alpha= pi*.2 ; %test_alpha= pi*rand ; %radians
%     I(:, :, 1)=(Ao.*cos (arb+test_phi)+ Ar*cos (arb)).^2 ;
%     I(:, :, 2)=(Ao.*cos (arb+test_phi-test_alpha)+ Ar*cos (arb)).^2 ;
    I(:, :, 1)=(Ao.*exp(i*(arb+test_phi))+ Ar*exp(i*(arb))).* ...
        conj((Ao.*exp(i*(arb+test_phi))+
Ar*exp(i*(arb))))+rand(blocksize,blocksize)*noise ;
    I(:, :, 2)=(Ao.*exp(i*(arb+test_phi-test_alpha))+
Ar*exp(i*(arb))).* ...
        conj((Ao.*exp(i*(arb+test_phi-test_alpha))+
Ar*exp(i*(arb))))-rand(blocksize,blocksize)*noise;

    fig=100;figure (fig);
    surf(X,Y,test_phi);hold all;
    xlabel('test_phi versus X,Y ');
end;
if jjj == 2
    figure (fig+1);
    surf(X,Y,I(:, :, 1)); %hold all;
    xlabel('I(:, :, 1) versus X,Y ');
% end;
    figure (fig+2);
    surf(X,Y,I(:, :, 2)); %hold all;
    xlabel('I(:, :, 2) versus X,Y ');
end;
%*****

%solving the problem?????
kend=200;
kendd=1;
ALPHA=zeros (kend,1);
AR=zeros (kend,1);
AO=zeros (blocksize,blocksize,kendd);
IO=zeros (blocksize,blocksize,kendd);
cPHI=zeros (blocksize,blocksize,kendd);
sPHI=zeros (blocksize,blocksize,kendd);
AOout=[99 99];
ALPHA(1)= pi*0.3;

%     AR(1)=1.0
AR(1)=mean (mean (I(:, :, 1)))^0.5;
k=1

```

```

while k < kend      %begin k loop
    dropline=1;
    k=k+1;
    p=I(:, :, 1)+I(:, :, 2)+(AR(k-1)^2)*2*cos(ALPHA(k-1));
%       q=((I(:, :, 1)+I(:, :, 2)-2*AR(k-1)^2).^2 + (I(:, :, 1)-I(:, :, 2)).^2
...
%           / (tan(ALPHA(k-1)/2))^2)/4
q=((I(:, :, 1)+I(:, :, 2)-2*AR(k-1)^2).^2 + (I(:, :, 1)-I(:, :, 2)).^2 ...
    / (tan(ALPHA(k-1)/2))^2)/4;

%       p2m4q=p.^2-4*q;
%       pmp2m4q=p-abs(p2m4q).^0.5;
p2m4q=p.^2-4*q;
pmp2m4q=p-abs(p2m4q).^0.5;
IO(:, :, kendd) = abs(pmp2m4q)/2;
AO(:, :, kendd) = (abs((p-abs(p.^2-4*q).^0.5)/2)).^0.5;

% is it best to calculate phi or cos(phi) ?????

%       PHIC=acos(( I(:, :, 1)-AO(:, :, k).^2-AR(k-1)^2)/(2*AR(k-
1).*AO(:, :, k)))
%       PHIS=asin((I(:, :, 2)-I(:, :, 1))/(2*AR(k-1).*AO(:, :, k)*sin(ALPHA(k-
1))))+...
%           (I(:, :, 1)-AO(:, :, k).^2-AR(k-1)^2)*tan(ALPHA(k-1)/2)/(2*AR(k-
1).*AO(:, :, k)))

    cPHI(:, :, kendd)=(( I(:, :, 1)-IO(:, :, kendd)-AR(k-1)^2)/(2*AR(k-
1).*AO(:, :, kendd)));
    temp1=(I(:, :, 2)-I(:, :, 1))/(2*AR(k-1).*AO(:, :, kendd)*sin(ALPHA(k-
1)));
    temp2= (I(:, :, 1)-IO(:, :, kendd)-AR(k-1)^2)*tan(ALPHA(k-
1)/2)/(2*AR(k-1).*AO(:, :, kendd));
    sPHI(:, :, kendd)=(temp1 +temp2);
if ~isreal(cPHI) | ~isreal(sPHI)
    dropline =9;
    disp(['cPHI or sPHI is complex at k=', num2str(k), '; canceled at
line ', num2str(dropline) ,Aname])
%       FinalAR=AR(k)
%       FinalAlpha=ALPHA(k)
    AOout=[mean(mean(AO)) mean(var(AO))];
%       return
dataout(jjj, :)= [k AOout AR(k-1) svolt(compare) svolt(jjj) ALPHA(k-1)
dropline xxx yyy]
break
end;
PHI=atan2(sPHI(:, :, kendd), cPHI(:, :, kendd));
%       figure(1);
%           rose(test_phi(:))
%           xlabel('test_phi rose ')
if ~isreal(PHI)
    dropline =6;
    disp(['PHI is complex at k =', num2str(k), ' canceled at line ',
num2str(dropline) ,Aname] )
    AOout=[mean(mean(AO)) mean(var(AO))];
dataout(jjj, :)= [k AOout AR(k-1) svolt(compare) svolt(jjj) ALPHA(k-1)
dropline xxx yyy]

```

```

break
end;
if jjj == 2

figure(fig+3);
set(fig+3,'Position',[1 scrsz(4)*.5 scrsz(3)*.3 scrsz(4)*.3])
    rose(PHI(:));
    xlabel('aPHI rose ');
end;
% % diffffphi=rem(test_phi,pi)-rem(aPHI,pi)
% diffffphi=exp(i*test_phi)-exp(i*PHI)
% diffff(k)=sum(sum(diffffphi))
% % if abs(diffff(k)) > 3*blocksize
% % disp(['PHI does not match canceled at line ',
num2str(dropline) ,Aname] )
% % cPHI(:, :, k)
% % sPHI(:, :, k)
% % return;
% % end;
% solve for IR c1 c2
% x = A\b
if rerunning == 99
    aamatrix=[blocksize^2 sum(sum(cos(test_phi)))
sum(sum(sin(test_phi)))
sum(sum(cos(test_phi)) sum(sum(cos(test_phi).^2))
sum(sum(cos(test_phi).*sin(test_phi)))
sum(sum(sin(test_phi)) sum(sum(cos(test_phi).*sin(test_phi)))
sum(sum(sin(test_phi).^2))];

bbmatrix=[sum(sum(I(:, :, 2)-Ao.^2))
sum(sum((I(:, :, 2)-Ao.^2).*cos(test_phi)))
sum(sum((I(:, :, 2)-Ao.^2).*sin(test_phi)))]];
xxmatrix=aamatrix\bmatrix;
end;
% % amatrix=[blocksize^2 sum(sum(cos(aPHI))) sum(sum(sin(aPHI)))
% % sum(sum(cos(aPHI))) sum(sum(cos(aPHI).^2))
sum(sum(cos(aPHI).*sin(aPHI)))
% % sum(sum(sin(aPHI))) sum(sum(cos(aPHI).*sin(aPHI)))
sum(sum(sin(aPHI).^2))]
% %
% % bmatrix=[sum(sum(I(:, :, 2)-IO(:, :, k)))
% % sum(sum((I(:, :, 2)-IO(:, :, k)).*cos(aPHI)))
% % sum(sum((I(:, :, 2)-IO(:, :, k)).*sin(aPHI)))]
amatrix=[blocksize^2 sum(sum(cPHI(:, :, kedd)))
sum(sum(sPHI(:, :, kedd)))
sum(sum(cPHI(:, :, kedd)) sum(sum(cPHI(:, :, kedd).^2))
sum(sum(cPHI(:, :, kedd).*sPHI(:, :, kedd)))
sum(sum(sPHI(:, :, kedd)) sum(sum(cPHI(:, :, kedd).*sPHI(:, :, kedd)))
sum(sum(sPHI(:, :, kedd).^2))];

bmatrix=[sum(sum(I(:, :, 2)-IO(:, :, kedd)))
sum(sum((I(:, :, 2)-IO(:, :, kedd)).*cPHI(:, :, kedd)))
sum(sum((I(:, :, 2)-IO(:, :, kedd)).*sPHI(:, :, kedd)))]];

xmatrix=amatrix\bmatrix;
IR(k)=xmatrix(1);

```

```

C1(k)=xmatrix(2);
C2(k)=xmatrix(3);
% x=pinv(amatrix)*bmatrix test matrix
% C1x=sum(sum(2*AO*AR*cos(test_alpha)));
% C2x=sum(sum(2*AO*AR*sin(test_alpha)));
% C2x_C1x=C2x/C1x;
% signC1(k)=sign(C1(k))
% signC2(k)=sign(C2(k))
if ~isreal(C2) | ~isreal(C1)
    dropline =5
    disp(['C2 or C1 is complex at k=',num2str(k),'; canceled at line ',
num2str(dropline) ,Aname] )
%     FinalAR=AR(k)
%     FinalAlpha=ALPHA(k)
%     test_alpha
AOout=[mean(mean(AO)) mean(var(AO))];
dataout(jjj,:)= [k AOout AR(k-1) svolt(compare) svolt(jjj) ALPHA(k-1)
dropline xxx yyy];
break
end;
ALPHA(k)=atan2(C2(k),C1(k));

AR(k)=IR(k)^0.5;
if ~isreal(ALPHA) | pi-abs(ALPHA(k)) < 0.001
    dropline =4
    disp(['ALPHA(',num2str(k),' ) is complex or pi; canceled at line ',
num2str(dropline) ,Aname])
%     FinalAR=AR(k)
%     FinalAlpha=ALPHA(k)
%     test_alpha
AOout=[mean(mean(AO)) mean(var(AO))];
dataout(jjj,:)= [k AOout AR(k-1) svolt(compare) svolt(jjj) ALPHA(k-1)
dropline xxx yyy]
break
end;
if abs(ALPHA(k-1)-ALPHA(k))< dALPHA
    dropline =2
AOout=[mean(mean(AO)) mean(var(AO))];
    dataout(jjj,:)= [k AOout AR(k) svolt(compare) svolt(jjj) ALPHA(k)
dropline xxx yyy]
    rerunning=0;
break
end;

    end; % end k loop

if dropline == 1
    dropline =3; AOout=[mean(mean(AO)) mean(var(AO))];
dataout(jjj,:)= [k AOout AR(k) svolt(compare) svolt(jjj) ALPHA(k)
dropline xxx yyy]
rerunning=0 ;
end;
end; % end rerunning ==99

Iout(:, :,1)=AO.^2 + AR(k)^2 + 2.*AO.*AR(k).*cPHI;

```

```

        Iout(:, :, 2)=AO.^2 + AR(k)^2 +
2.*AO.*AR(k).*(cPHI*cos(ALPHA(k))+sPHI*sin(ALPHA(k)));
        fig=1000+jjj;
%           if sam??? > 2
%           close(fig);
%           end;
        figure(fig);
        set(fig, 'Position', [1 scrsz(4)*.03 scrsz(3)*.4
scrsz(4)*.4])
        plot(X(1, :), I(blocksize/2, :, 1), 'b');hold all;
        plot(X(1, :), I(blocksize/2, :, 2), 'r');hold all;
        plot(X(1, :), Iout(blocksize/2, :, 1), '-.g');hold all;
        plot(X(1, :), Iout(blocksize/2, :, 2), '-.k');hold all;
        xlabel([num2str(jjj), 'I(1, :, 1)blue    I(1, :, 2)red
Iout(1, :, 1)green  Iout(1, :, 2)black'])
end; % end jjj ~= compare
end; % end of jjj loop
if step ==1
    datahold=zeros(xxtimes*yytimes,M,10);
end;
    datahold(step, :, :) = dataout;
    xxyylabell= [step];
fig=999;
figure(fig);
set(fig, 'Position', [1 scrsz(4)*.03 scrsz(3)*.4 scrsz(4)*.4])
plot(dataout(:, 6), dataout(:, 7), '-.d');hold all;
xlabel([colorr, ' Phase versus svolt']);

legend (num2str(xxyylabell), 'Location', 'NorthWest');

    end; % end yy loop
end; % end xx loop
datareal=real(datahold)
toc

droplines={'1 is default'; ...
['2 diamond is end at less than dALPHA ', num2str(dALPHA), ' line
369'];...
'3 + is end at k loop line 379';...
'4 o Alpha is complex or pi line 359';...
'5 * C2 or C1 is complex line 346';...
'6 ^ PHI is complex line 286';...
'9 x cPHI or sPHI is complex line 272'}

% Plot of only "GOOD" DATA
set(0, 'Units', 'pixels')
scrsz = get(0, 'ScreenSize')
counter=size(datareal)
fig=998;
figure(fig);
set(fig, 'Position', [1 scrsz(4)*.03 scrsz(3)*.4 scrsz(4)*.4]);
goodsum=zeros(1, counter(2));
goodn=zeros(1, counter(2));
for x=1:counter(1)

```



```

for y=1:counter(2)
switch datareal(x,y,8)
case 2
mark='dg';
case 3
mark='+r';
case 4
mark='or';
case 5
mark='*r';
case 6
mark='^r';
case 9
mark='xr';
otherwise
mark='sk';
end;

plot(datareal(x,y,6),datareal(x,y,7),mark);hold all;
if datareal(x,y,8) == 2
goodsum(y)=goodsum(y)+datareal(x,y,7);
goodn(y)=goodn(y)+1;
end;

end;

end;
goodphase=goodsum(:)./goodn(:)
plot(datareal(1,:,6),goodphase,'hk','MarkerSize',15);hold
all;
xlabel('Volts AC @ 60 HZ','FontWeight','bold');
ylabel('Radians','FontWeight','bold');
title('Relative Phase versus Input Voltage','FontWeight','bold');
%
ARsum=[];AOsum=[];ARcount=0;
for x=1:counter(1)

for y=1:counter(2)
if datareal(x,y,8)==2
ARsum=[ARsum datareal(x,y,4)];
AOsum=[AOsum datareal(x,y,2)];
ARcount=ARcount+1;
end;
end;
end;
ARcount=ARcount
AOmean=mean(AOsum)
ARmean=mean(ARsum)
AR_AOratio=ARmean/AOmean
Transmission_AR=ARmean^2/(ARmean^2+AOmean^2)
Transmission_AO=AOmean^2/(ARmean^2+AOmean^2)

```