MILLIMETER-WAVE MONOLITHIC SCHOTTKY DIODE-GRID PHASE SHIFTER

Thesis by

Wayne W. Lam

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1987

(Submitted January 31, 1987)

TO MY PARENTS

Acknowledgements

I would like to express my deepest thank to Professor David Rutledge, my thesis advisor, for having given me the opportunity to be a member of his Millimeter-Wave Integrated Circuits Group. His ideas, encouragement and guidance made this work possible, and his kindness and understanding helped me grow. It has been my good fortune to have met him.

I am grateful to Professor Dean Neikirk for showing me the art of fabricating microbolometers. I am indebted to Dr. Chung-en Zah for teaching me the techniques of diode fabrication.

Also, I would like to thank Professors William Bridges, Marc-Aurele Nicolet, and Amnon Yariv at Caltech, and to Professor Neville Luhmann, Jr., at UCLA, for the use of their equipment and facilities. Especially, I would like to extend my thanks to Howard Chen for supplying me the epitaxial wafers for my research, Frank So for doing proton bombardments, Christina Jou for helping me in diode-grid measurements. In addition, I appreciate Dr. Thomas Kuech at IBM for sending me MOCVD wafers, Kjell Stolt at TRW and Lars Eng at JPL for growing the epitaxial wafers, and Bob Rush at Hughes for proton implantation.

Furthermore, I would like to thank Joyce Liddell for her secretarial assistance, Wyman Williams and Rick Compton for sharing their computer expertise, and other members of the group for their friendship and support.

I am grateful for the financial support of the Army Research Office Fellowship and the AMOCO Foundation Fellowship.

And finally, I am most thankful to my parents, Hing Shing and Kor Han for giving me a better chance in life and my brother, Lawrence, for getting me interested in going to college rather than to the beach.

Millimeter-Wave Monolithic Schottky Diode-Grid Phase Shifter

Abstract

Many applications at millimeter wavelengths require fast electronic phase shifters. In this study, the design of diode-grid phase shifters is presented, the fabrication of diode-grids on monolithic gallium-arsenide substrates is demonstrated, and the measurement of these grids is discussed. A new computer-aided design tool is developed to provide an interactive environment for design and to form a basis for comparing theory and experimental results. Diode-grids have been fabricated on 2 cm by 3 cm gallium-arsenide wafers with 2000 aluminum Schottky diodes. A novel small aperture reflectometer is computerized to use a wave-front division interference technique to measure the reflection coefficient of the grids. A 70° phase shift with a 6.5-dB loss was measured at 93 GHz when the bias on the diode-grid was changed from −3 V to +1 V.

TABLE OF CONTENTS

Chapter 1

Introduction

Unique features of millimeter waves have attracted a growing interest in the wavelength region from 1 mm to 10 mm. Millimeter waves offer broader bandwidths, higher resolution and smaller component size than microwaves, and provide better penetration of fog and dust than infrared [1]. In radio astronomy, the measurement of molecular resonance lines, which occur significantly in the millimeter range, have provided important physical insights into the state of interstellar clouds [2]. In plasma diagnostics, the measurement of electron density and temperature profile, and fluctuation have increased the understanding of particle confinement mechanisms in a fusion plasma [3]. Thermography at millimeter wavelengths has been used in tumor detection [4]. Other applications including active radiometry [4], satellite communication [5], and remote-sensing of the earth's surface [1] are beginning to receive more attention.

As the number of applications increases, demands for new and improved systems follow. Currently standard components available for millimeter applications are based on metal cavities, waveguides, and horns. Although they are adequate for small systems, they become quite expensive to use in large systems because intensive labor is required in machining these parts. In situations where only a limited amount of space is available, they can be very difficult to manage, since their physical dimensions tend to be much bigger than a wavelength. These are the driving forces that have led to the recently increased efforts in research and development of monolithic integrated circuits for millimeter waves. Special issues on this subject have appeared in the IEEE Transactions [6,7]. Recently, Stiglitz [9] presented a special report on the topic of gallium-arsenide technology and microwave and millimeter-wave monolithic integrated circuits for 1987.

1.1 Applications of Diode-Grids

Much of the current research activities in monolithic integrated circuits for millimeter waves tend to revolve around a variety of planar transmission-line structures and dielectric waveguides. Recently a different structure based on integrating solid-state devices into a periodic grid has emerged. Rutledge and Schwarz demonstrated a multi-mode detector array by integrating microbolometers into a periodic grid [10]. Tong *et al.* built a two-dimensional tracking array, also with microbolometers and a periodic grid [11]. Figure 1.1 shows a periodic grid loaded with Schottky varactor-diodes, hence the name diode-grid. Designs based on the diode-grid for electronic beam-steering and frequency multiplication had been proposed [12], and the fabrication of a diode-grid was demonstrated on a 2 cm by 2 cm gallium-arsenide wafer with 2000 diodes [13].

Periodic grids offer many advantages. They present a planar geometry that is both simple and compact; therefore, it has a tremendous capacity for interconnecting thousands of solid-state devices on a single substrate. The periodic grid lends itself to a variety of high-power applications, since power is distributed among a large number of electronic devices throughout a planar surface. They are exciting because they open up a new area of monolithic integrated circuits for plane waves. This provides an extra degree of freedom to the circuit designer. Basically, the longitudinal dimension can be used for guiding high-frequency signals and feeding electromagnetic energy to the devices, while low-frequency control signals and bias can be routed in the transverse dimension. The system design is analogous to an optics design. Furthermore, the diode-grid approach is compatible with the semiconductor fabrication technology. This leads to lower cost, smaller size, and more reliable components. No transmission lines or waveguides are used. This makes fabrication simpler and losses lower.

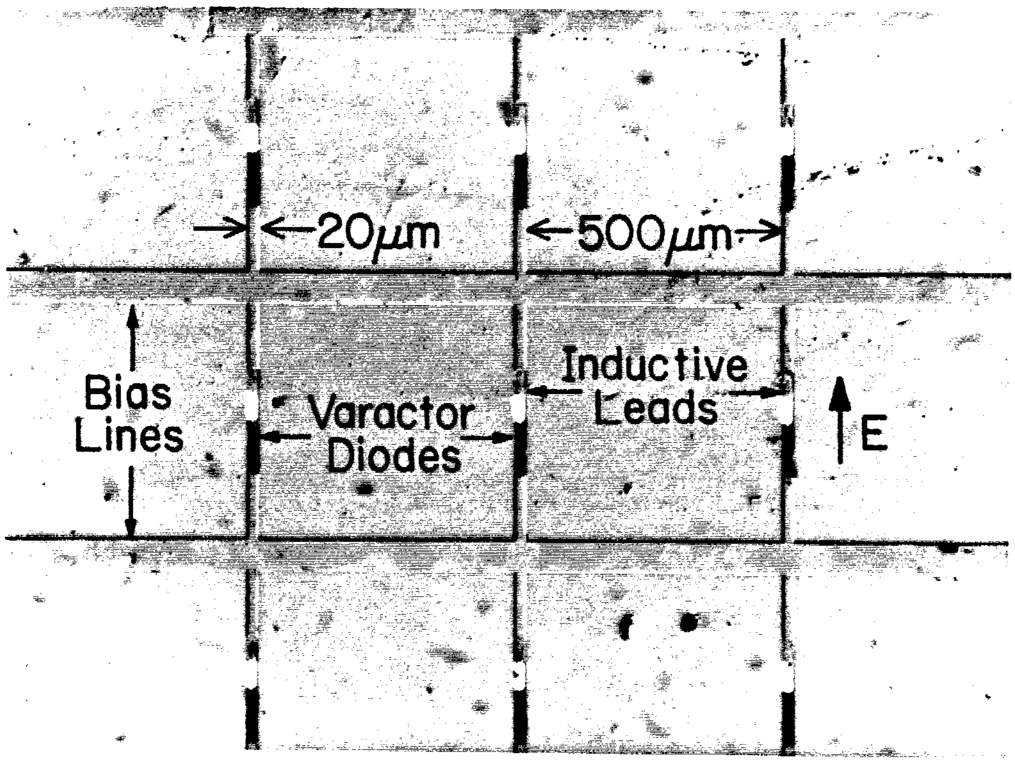


Figure 1.1. Part of a periodic diode-grid fabricated on a gallium-arsenide wafer.

One of the applications of the diode-grid is in electronic beam-steering. Currently, beam-steering plays a vital role in advanced radars that track and image multiple objects simultaneously. The key element that enables a beam of radiation to be steered at electronic speeds is the phase shifter. Typically, many thousands of phase-shifting elements are required. Conventional phase-shifters based on microwave hybrids of striplines and waveguides lead to high cost and system interconnecting complexity as the wavelength approaches 1 mm. Recently Horn *et al.* [14] demonstrated an electronically modulated line scanning antenna. However, many applications require more gain than a line source can provide. The use of variable-permittivity media for phase-shifting is an intriguing alternative possibility [15,16]. Figure 1.2 shows a varactor-diode grid design for electronic beam-steering. In the beam-steering array, the incident beam reflects off the programmable diode-grid phase-shifting surface, where changing the dc bias on the diodes changes the reactance, and this controls the phase of the reflected waves. A linear variation of the phase across the aperture sets the direction of the reflected beam. In addition, a quadratic variation of the phase across the aperture focuses the reflected beam. No transmission lines or waveguides are required. This architecture makes the system design simpler and the fabrication cost lower. Since the power is distributed among all the diodes, the power handling capability can be designed specifically for a particular application by choosing the array size properly.

Another application of the diode-grid is in harmonic power generation. As the wavelength approaches 1 mm, the varactor multiplier plays an important role in providing local power to heterodyne receivers because other sources present many undesirable features. Tubes require cumbersome and dangerous high-voltage supplies, Impatt oscillators are generally too noisy, and Gunn oscillators are not able to provide sufficient power at frequencies above 100 GHz. Recently, Archer [17]

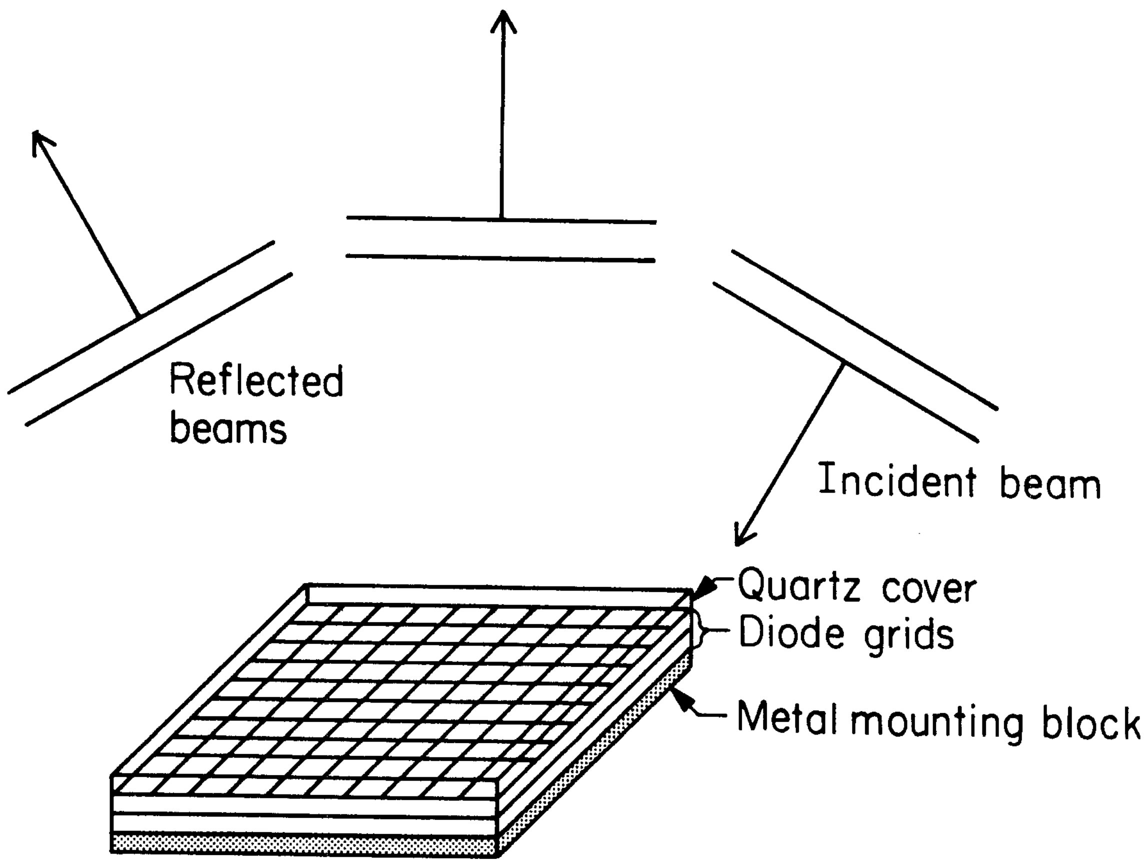


Figure 1.2. Electronically programmable beam-steering array.

summarized state-of-the-art performances for millimeter wavelength frequency multiplier. They are primarily based on using a single whisker-contacted varactor diode chip in a cross-waveguide configuration. Although very high conversion efficiency has been obtained, they are basically limited to milliwatts of output power. The fundamental limitation is that only a few varactors can be used simultaneously in a practical manner. However, power can be increased significantly when thousands of varactors are combined together in a suitable manner for synchronous operation. Figure 1.3 shows a second harmonic power-combiner that uses an array of nonlinear capacitors to generate and spatially combine power at the harmonic frequencies. In this frequency doubler design, power at the fundamental frequency enters through the input filter, arrives at the varactor-diode grid, and pumps the nonlinear capacitance of the diodes to generate power at harmonics. The second harmonic is spatially combined and transmitted through the output filter.

Loading a grid with negative resistance diodes offers the possibility of dc-to-rf power conversion that is analogous to a laser. Currently, oscillators with single electronic device such as Gunn or IMPATT diode are highly developed. However, they are capable of providing continuous-wave power only from about 500 mW at 40 GHz to 10 mW at 230 GHz [18]. Many applications in radars, imaging arrays, and heterodyne receivers require much more power than this. Although a number of power-combining circuits have been demonstrated including chip-level, circuit-level, and spatial power-combining [19], they do not take full advantage of what the solid-state semiconductor technology can offer. Recently Wandinger and Nalbandian demonstrated a dual oscillator quasi-optical power-combining resonator at 60 GHz with dielectric antennas [20]. However, extending this approach to higher-level combining would be expensive and difficult. Mink investigated theoretically a very interesting distributed source planar array

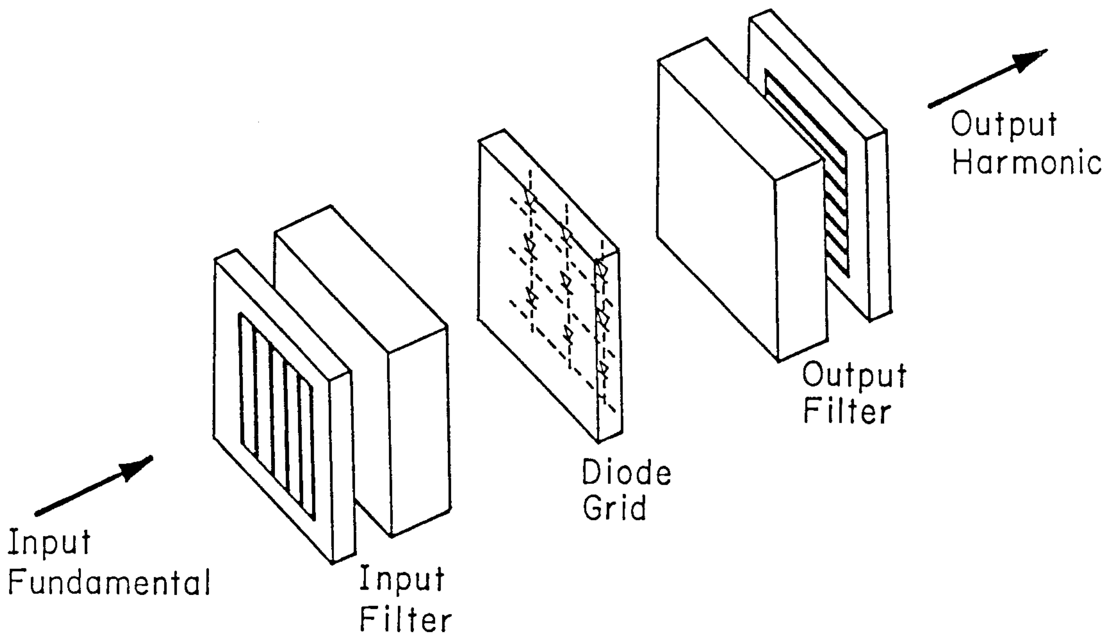


Figure 1.3. Second harmonic power-combiner.

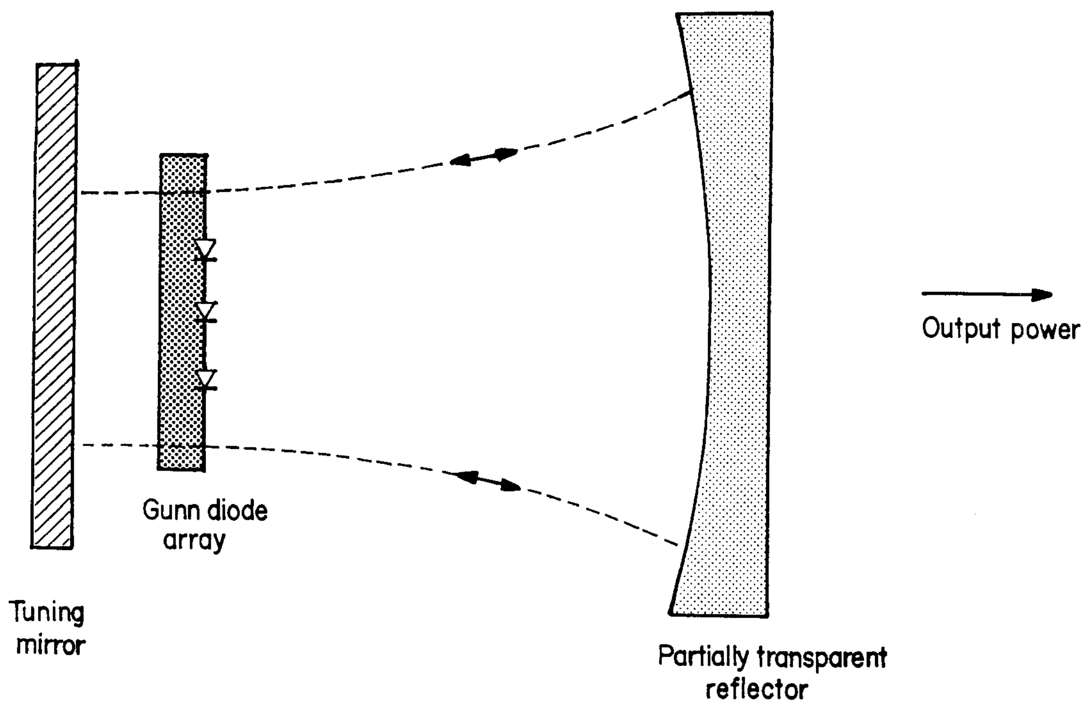


Figure 1.4. Gunn-diode resonator.

resonator [21]. Figure 1.4 shows a Gunn diode-grid resonator. Power generated by the Gunn diodes are combined in a semi-confocal cavity. A metal mirror is used as a tuning short for matching the impedance of the Gunn diode. Power can be coupled out through a partially transparent mirror. In addition, a union of the Gunn diode-grid resonator and the harmonic power-combiner would be an exciting all solid-state alternative for high-power local oscillator source.

Another interesting application of the diode-grid is in signal detection. Recently, Zah *et al.* [22] demonstrated a one-dimensional Schottky-diode imaging array at 90 GHz. Figure 1.5 shows a typical lens-coupled optical system that has been used in imaging experiments today [23,24]. The idea is to focus energy onto an imaging array at the focal plane, so that an image of the object can be constructed at electronic speed. Figure 1.6 is a close-up view of a two-dimensional imaging array based on the diode-grid approach. The idea is to use the horizontally connected diodes in the back layer as tuning elements for the vertically connected diodes in the front layer. The intersection of a column and a row defines a pixel element. The amplitude at each pixel can be measured from the column with the proper dc bias applied to the corresponding row. Furthermore, being able to ascertain informations from each unit cell of a finite periodic structure raises the possibility of studying edge effects due to finite periodicity. This should provide important insight into future works concerning circuit interactions between diode-grids on different planes.

Finally, applications of the diode-grid, analogous to holography and nonlinear optics, also appear possible. In optical phase conjugation, a nonlinear medium is used to generate the complex conjugate of a wave. This is useful in real-time imaging through a phase distorting medium [25]. The technique of using a nonlinear surface for generating a phase-conjugated wave has been demonstrated at optical wavelengths [26]. Figure 1.7 shows how this can be done with a varactor-

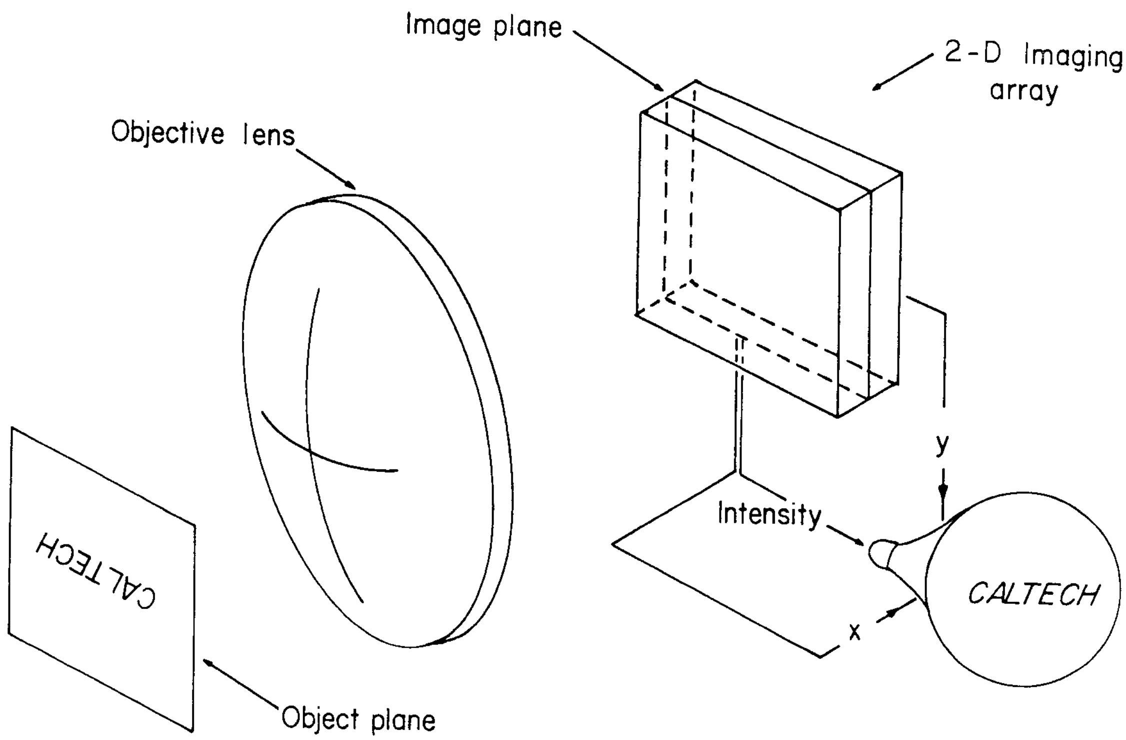


Figure 1.5. Two-dimensional imaging system.

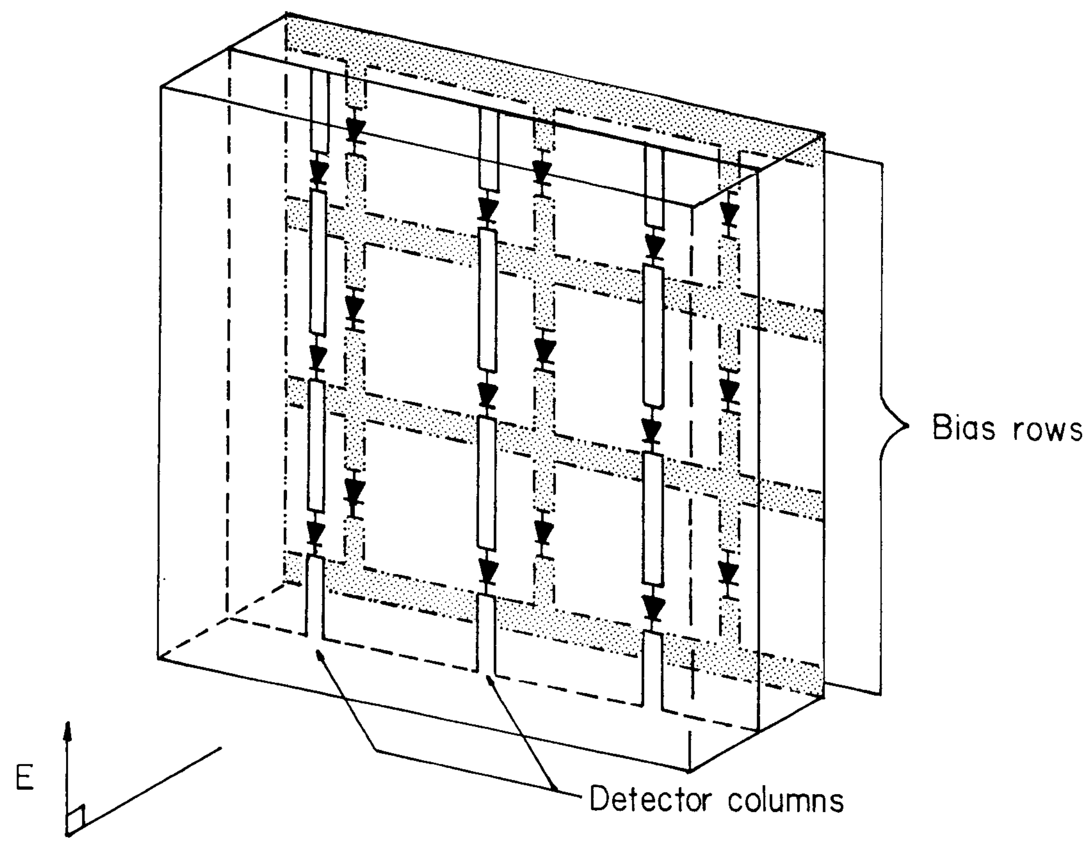


Figure 1.6. Two-dimensional imaging array.

diode grid at millimeter wavelengths. The idea is to mix the signal-beam with a normally incident pump-beam on the surface of the varactor diode-grid, so that a hologram in the form of reflectivity modulation is developed. During this mixing process, the phase-conjugated beam is generated and the replica of the signal beam is reproduced as the conjugated beam evolves through the phase-distorting medium. Also signal amplification can be obtained.

1.2 Overveiw of Thesis

The numerous potential applications of the diode-grid is the motivation behind this research. The purpose of this thesis is to lay down the groundwork and to demonstrate feasibility. Chapter 2 presents design considerations, a model of the diode-grid, and computer simulations of the reflection phase-shifter for electronic beam-steering. The approach is based on a transmission-line equivalent circuit. In designing the grid structures, together with the substrate, dielectric slabs, filters and mirrors, a computer-aided design program was developed to provide an interactive environment for the user and to form a basis for comparing theoretical and experimental results. The software documentation is given in the appendix.

Chapter 3 describes the design of a truncated hyperabrupt doping distribution for making a Schottky-barrier varactor diode. Essential parts of the diode fabrication are discussed. Zah's self-aligning process as described by Zah *et al.* [22] is used to fabricate the aluminum Schottky diodes. Monolithic diode-grids have been fabricated on 2 cm by 3 cm gallium-arsenide wafers with 2000 varactor diodes. Although the diode fabrication yield as high as 98 % has been achieved, the remaining bad diodes, which tend to be short circuits, usually render the grid useless. Therefore, a liquid-crystal detection technique was developed to identify the shorted diodes, which are subsequently removed by an ultrasonic probe.

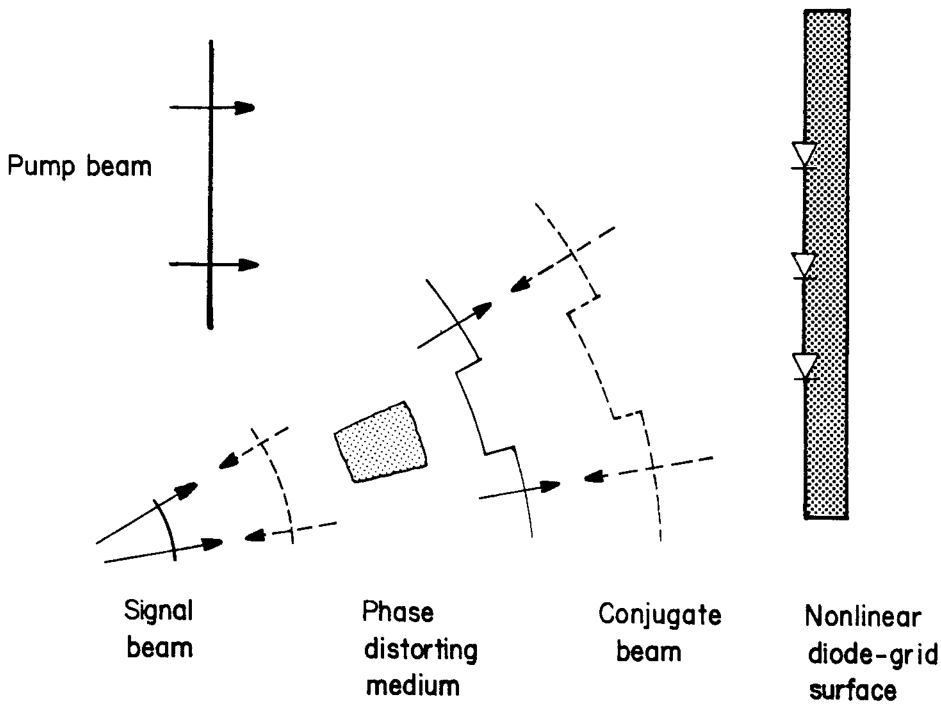


Figure 1.7. Phase-conjugating diode-grid.

The diode series resistance is calculated from the current-voltage measurement at DC, and the doping distribution is extracted from the capacitance-voltage measurement at 1 MHz. Detailed procedures are given in the appendix.

Chapter 4 begins with a survey of possible existing experimental methods for testing the diode-grid and concludes that none was practical for testing our diode-grid samples because the sample sizes are typically small and irregular in a laboratory environment. Therefore, a novel small aperture quasi-optical reflectometer was developed to meet this need. It uses a wave-front division interference technique to measure the reflection coefficient of a surface. The technique was first applied to measure the reflection coefficient of bismuth on quartz. The purpose was to get an indication of its validity and limitations. Then it was used for testing the diode-grids. Results measured at 90 GHz are curve-fitted with a theory based on a transmission-line equivalent circuit model. The best-fit grid parameters are compared with parameters measured at low frequency.

References

[1] P. Bhartia and I. J. Bahl, *Millimeter Wave Engineering and Applications*, **Chap. 1,** pp. 1–8, John Wiley & Sons, Inc., New York, 1984.

[2] C. H. Townes, "The Challenge of Astronomy to Millimeter-Wave Technology," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-24,** pp. 709–711, 1976.

[3] N. C. Luhmann, Jr. "Instrumentation and Techniques for Plasma Diagnostics: An Overview," *Infrared and Millimeter Waves*, **Chap. 1,** pp. 1–66, K. J. Button, ed., Academic Press, Inc., New York, 1983.

[4] J. Edrich, "Centimeter and Millimeter-Wave Thermography–A Survey on Tumor Detection," *J. Microwave Power*, **Vol. 14,** pp. 95–104, 1979.

[5] W. J. Wilson, R. J. Howard, A. C. Ibbott, G. S. Parks, and W. B. Ricketts, "Millimeter-Wave Imaging Sensor," *IEEE Trans. on Microwave Theory and Tech.* **MTT–34,** pp. 1026-1035, 1986.

[6] K. Miyanchi, "Millimeter-Wave Communication," *Infrared and Millimeter Waves*, **Chap. 1,** pp. 1–94, K. J. Button, ed., Academic Press, Inc., New York, 1983.

[7] Special Issue on Microwave and Millimeter-Wave Integrated circuits. *IEEE Trans. on Microwave Theory and Tech.* **MTT-26,** pp. 705–843, 1978.

[8] Special Issue on Millimeter-Waves. *IEEE Trans. on Microwave Theory and Tech.* **MTT-31,** pp. 89–238, 1983.

[9] M. R. Stiglitz, "GaAs Technology and MIMIC, 1987," *Microwave Journal*, **September,** pp. 42–66, 1986.

[10] D. B. Rutledge and S. E. Schwarz, "Planar Multimode Detector Arrays for Infrared and Millimeter-wave Applications," *IEEE J. Quantum Electronics*, **QE-17,** pp. 407–414, 1981.

[11] P. P. Tong, D. P. Neikirk, D. Psaltis, K. Wagner, and P. E. Young, "Tracking

Antenna Arrays for Near-Millimeter Waves," *IEEE Trans. on Antennas and Propagat.*, **AP-31**, pp. 512–515, 1983.

[12] W. W. Lam, C. F. Jou, N. C. Luhmann, Jr., and D. B. Rutledge, "Diode Grids For Electronic Beam Steering And Frequency Multiplication," *Int. J. of Infrared and Millimeter Waves*, **Vol. 7**, pp. 27–41, 1986.

[13] W. W. Lam, H. Chen, D. B. Rutledge, C. F. Jou, N. C. Luhmann, Jr., "Progress in Diode Grids for Electronic Beam Steering and Frequency Multiplication," *11th Int. Conf. on Infrared and Millimeter Waves*, pp. 177–179, Pisa, Italy, Oct. 1986.

[14] R. E. Horn, H. Jacobs, K. L. Klohn, and E. Freibergs, "Single Frequency Electronic-Modulated Analog Line Scanning Using a Dielectric Antenna," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-30**, pp. 816–820, 1982.

[15] H. Buscher, "Electrically Controllable Liquid Artificial Dielectric Media," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-27**, pp. 540–545, 1979.

[16] R. H. Park, "Radant Lens: Alternative to Expensive Phased Arrays," *Microwave Journal*, **September**, pp. 101–105, 1980.

[17] J. W. Archer, "Low-Noise Heterodyne Receivers for Near-Millimeter-Wave Radio Astronomy," *Proceedings of the IEEE*, **vol. 73**, pp. 109–130, 1985.

[18] J. C. Wiltse, "Introduction and Overview of Millimeter Waves," *Infrared and Millimeter Waves*, **Chap. 1**, pp. 1–17, K. J. Button, ed., Academic Press, Inc., New York, 1981.

[19] K. Chang and C. Sun, "Millimeter-Wave Power-Combining Techniques," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-31**, pp. 91–107, 1983.

[20] L. Wandinger and V. Nalbandian, "Millimeter-Wave Power Combiner Using Quasi-Optical Techniques," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-31**, pp. 189–193, 1983.

[21] J. W. Mink, "Quasi-Optical Power Combining of Solid-State Millimeter-Wave

Sources," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-34**, pp. 189–193, 1986.

[22] C. Zah, D. Kasilingam, J. S. Smith, D. B. Rutledge, T. Wang, and S. E. Schwarz, "Millimeter-Wave Monolithic Schottky Diode Imaging arrays," *Int. J. of Infrared and Millimeter Waves*, **Vol. 6**, pp. 981–997, 1985.

[23] D. P. Neikirk, D. B. Rutledge, M. S. Muha, H. Park, and C. X. Yu, "Far-Infrared Imaging Antenna Arrays," *Appl. Phys. Lett.*, **Vol. 40**, pp. 203–205, 1982.

[24] D. B. Rutledge, D. P. Neikirk and D. P. Kasilingam, "Integrated-Circuit Antennas," *Infrared and Millimeter Waves*, **Vol. 10**, chap. 1, pp. 1–90, K. J. Button, ed., Academic Press, Inc. New York, 1983.

[25] A. Yariv, "Phase Conjugate Optics and Real-Time Holography," IEEE *J. of Quantum Electronics*, **QE-14**, pp. 650–660, 1978.

[26] B. Ya. Zel'dovich, N. F. Pilipetsky, and V. V. Shkunow, "Introduction to Optical Phase Conjugation," *Principles of Phase Conjugation*, Springer-Verlag, New York, 1984.

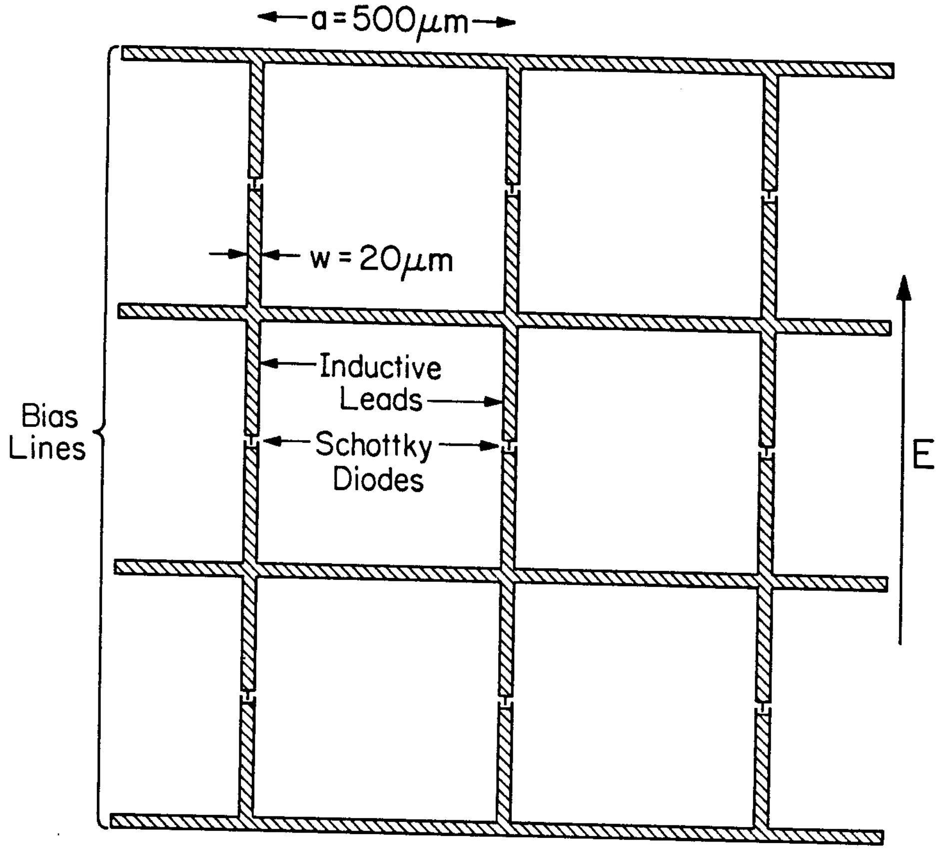
Chapter 2

Design and Analysis of Diode-Grid Phase Shifter

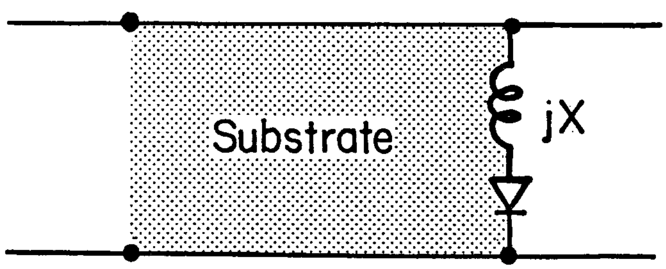
Many types of periodic grids have been used in infrared and millimeter-wave applications, including the Jerusalem-cross for band-reject filters [1], discs for artificial dielectrics [2], inductive wires for polarizers [3], inductive and capacitive strips for polarization independent beam splitters [4] and multiplexers [5], and metal meshes for output couplers for lasers [6]. The periodic grid that we use in designing our diode-grid phase shifter is a square mesh of metal strips on a gallium-arsenide substrate as shown in figure 2.1a. The vertical strips add inductance to cancel the diode capacitance. The horizontal strips provide the bias but should not otherwise affect the circuit. Design considerations include the grid period, angle of incidence, and dielectric constant. The design approach is based on an equivalent circuit model and the transmission-line theory. In designing the diode-grid phase shifter for electronic beam-steering, we developed a computer-aided design program to provide an interactive environment for the designer and to form a basis for comparing theoretical and experimental results.

2.1 Diode-Grid Model

We model the diode-grid with an equivalent circuit based on the transmission-line theory because it is relatively easy to incorporate both the diode model and supporting substrate into the analysis. Figure 2.1b shows a simple model of the diode-grid on a dielectric slab. The grid is represented by an inductor in series with a diode, and the substrate is represented by a section of transmission line with a characteristic impedance equal to the wave impedance in the dielectric. The inductance-per-unit length of the metal strip for normal incidence is given



(a)



(b)

Figure 2.1. (a) Grid dimensions for a 90 GHz programmable phase shifter. The incident electric field is assumed to be vertically polarized. (b) Transmission-line model of an inductive grid, loaded with diodes, and supported by a substrate.

by a quasi-static approximation,

$$L = \frac{\mu_o}{2\pi} \ln \left[\csc \left(\frac{\pi w}{2a} \right) \right], \tag{2.1}$$

where w is the strip width, a is the grid period, and μ_o is the magnetic permeability. MacFarlane [7] derived this formula based on conformal mapping of an inductive grating in a parallel plane metal waveguide. It does not take into account the angle of incidence, the polarization, the effect of the dielectric interface, the parasitic capacitance across the diode, or the effect of the horizontal cross strip. These effects have been considered in the literature [8,9]. Since they amount to a correction of less than 10 %, they have been neglected in our initial designs of the diode-grid. This enables us to see the effects of design changes faster and therefore to get a quicker turn-around time in doing the design of the diode-grid phase shifter.

The grid period, a, should be somewhat smaller than a substrate wavelength to avoid exciting substrate modes. We can decide how much smaller by considering figure 2.2, which shows the spatial frequencies of the grid. The spatial frequencies for the incident radiation lie within a circle of radius $1/\lambda_o$ centered at the origin. This radiation excites currents, which, because of the periodic nature of the grid, have spatial frequencies that lie within similar circles centered on the reciprocal lattice points of the grid. The reciprocal lattice is a square lattice, with a period of $1/a$. The spatial frequencies for the substrate modes lie within the doughnut-shaped regions that are also centered on the reciprocal lattice points. The inner radius of the doughnuts is $1/\lambda_o$ and the outer radius is n/λ_o, where n is the refractive index of the substrate. To avoid exciting substrate modes, the small circles should not intersect the doughnuts. This means the grid period

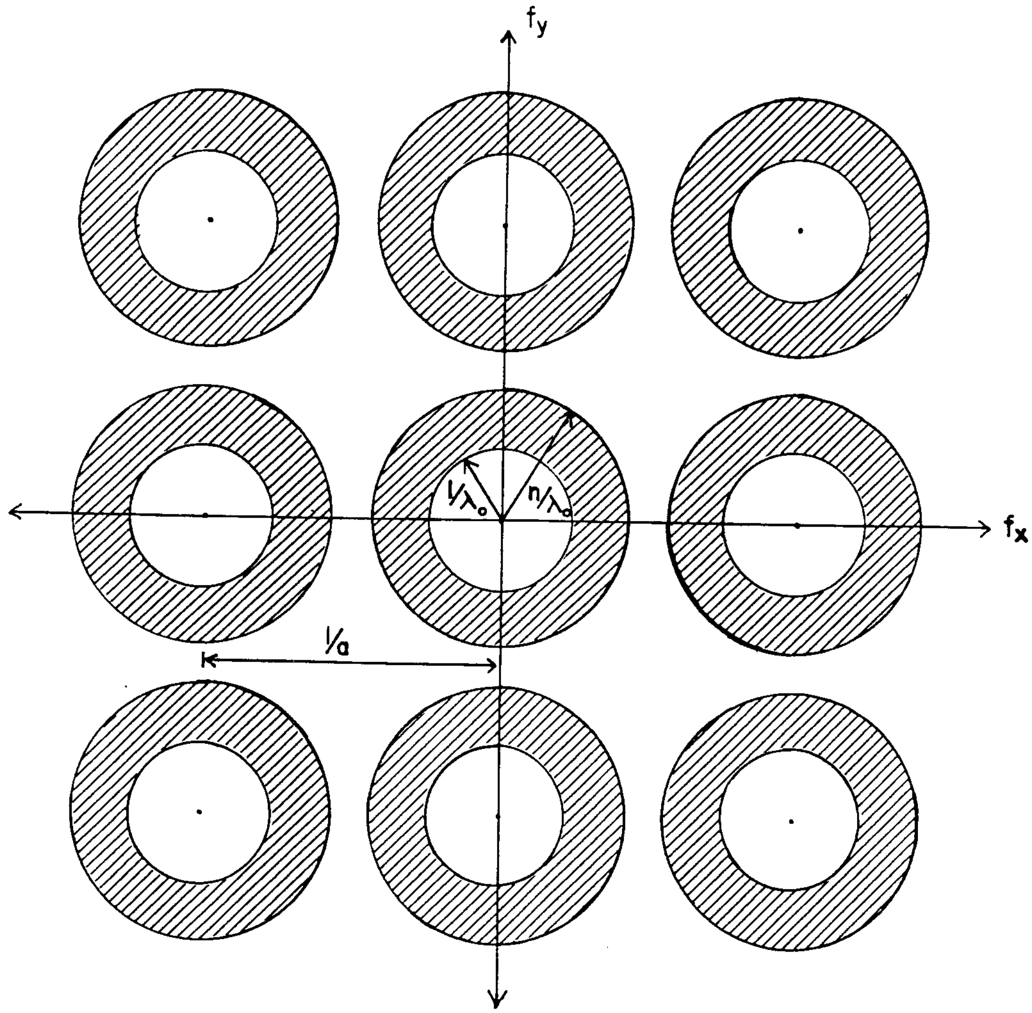


Figure 2.2. Reciprocal lattice of a square grid for considering the excitation of substrate modes.

should satisfy

$$a < \frac{\lambda_o}{(n+1)}. \tag{2.2}$$

For gallium-arsenide, which has a refractive index of 3.6, the grid period should be less than $0.22\lambda_o$.

2.2 Diode-Grid Phase Shifter

Figure 2.3 shows the diode-grid phase shifter design. It consists of a fused-quartz cover, two diode-grids, and a metal mirror. The circuit is analogous to Garver's microwave phase shifter [10]. The quartz layer acts as a protective cover as well as an impedance transformer. The metal mirror prevents radiation from escaping and serves as a heat sink and mechanical support. Another inherent feature of this design is that the mirror also shorts out the second harmonic at the diode-grids. This reduces conversion losses to the second harmonic; therefore, it is more attractive for high power-operation.

Figure 2.4a shows the transmission-line model. The mirror is electrically an open-circuit, because it is a quarter-wavelength behind the back diode-grid. At the front diode-grid, the back grid appears electrically as a parallel load, but with the impedance inverted (figure 2.4b). The total normalized reactance X_t is the parallel combination of jX and $1/jX$, or

$$X_t = \frac{X}{(1-X^2)}. \tag{2.3}$$

X_t ranges from $-\infty$ as X approaches -1, to $+\infty$ as X approaches $+1$. This allows a full 360° phase shift as the normalized grid reactance goes from -1 to $+1$. The grid reactance, X, is normalized relative to the characteristic impedance of the substrate. In gallium-arsenide this corresponds to a grid reactance sweep between $-107\,\Omega$ to $+107\,\Omega$. For the grid dimensions in figure 2.1, the inductive

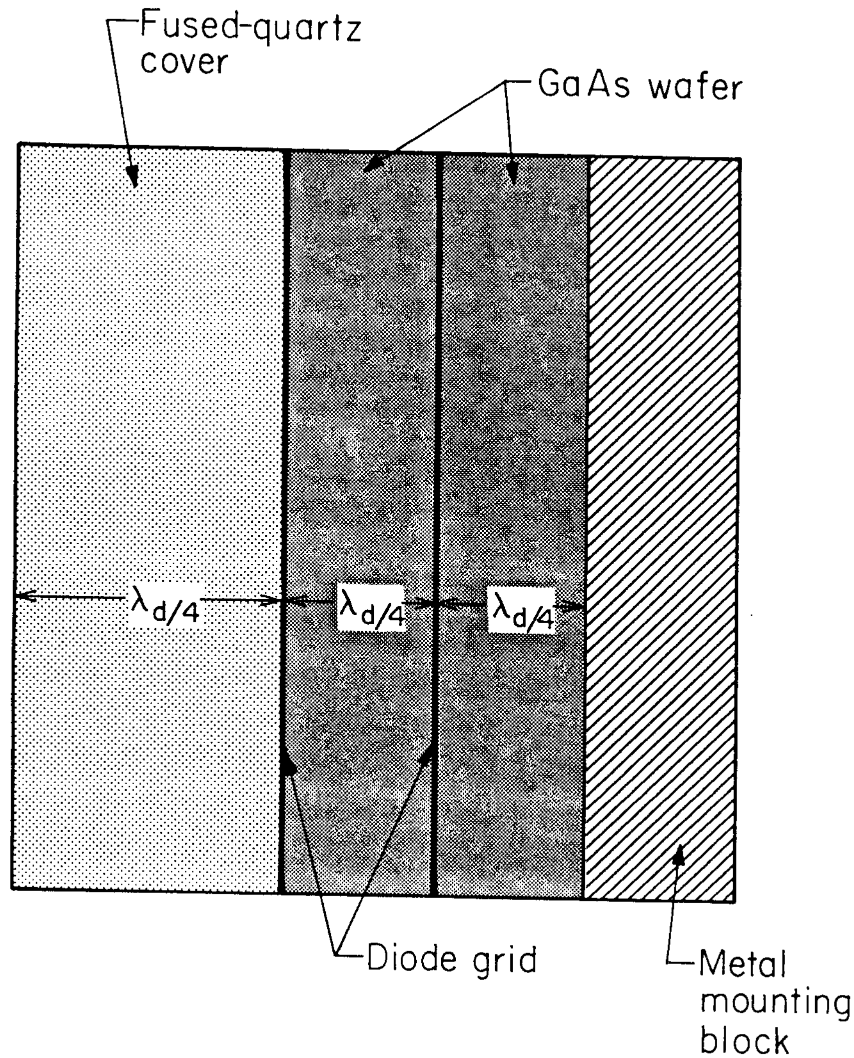
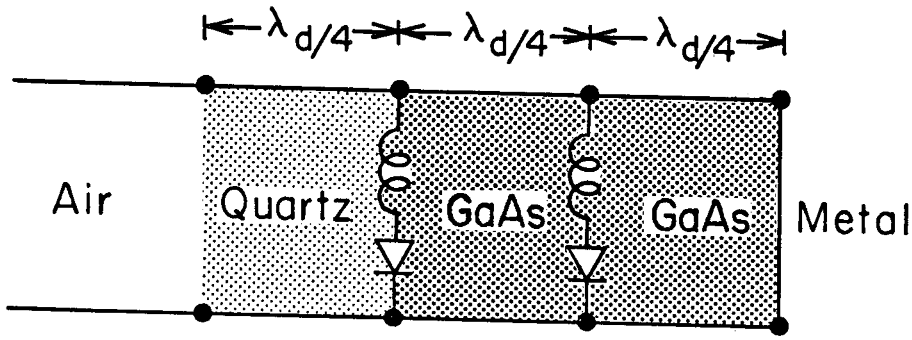
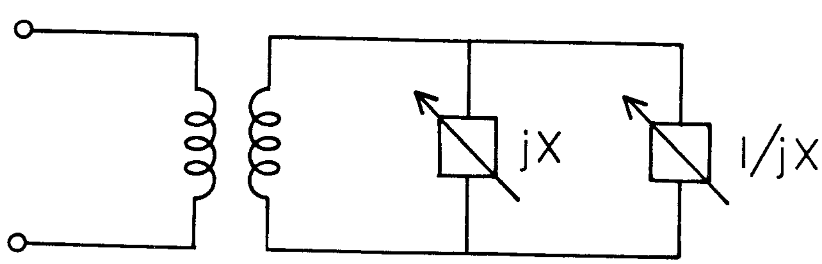


Figure 2.3. Side view of the programmable diode-grid phase shifter for electronic beam steering.



(a)



(b)

Figure 2.4. (a) Transmission-line model of the diode-grid phase shifter. (b) An idealized equivalent circuit for the diode-grid phaser shifter.

reactance due to the strip is 160 Ω. This means that the diode capacitance should vary from 7 fF to 35 fF. This type of capacitance ratio has been demonstrated with the hyperabrupt junction Schottky varactor diode [11].

The design allows a spatial phase variation in one dimension only. Two-dimensional phase variation can be achieved by biasing the diodes individually rather than row by row, or by cascading two such phase shifters. The design also does not control sidelobes because there is no adjustment for amplitudes. However, it should be possible to reduce sidelobes by tapering the radiation feeding the array with an externally designed collimating lens.

2.3 Computer-Aided Design and Analysis – TRAP

Since calculations of reflection modulus and phase of multi-layerd media are tedious and time consuming, we have developed a computer-aided design program that provides an interactive environment for the user to design his circuits and to compare the theoretical and experimental results. TRAP (transmission, reflection, absorption, and phase) was developed to analyze the square grid, together with the substrate, dielectric slabs, filters, and mirror. It is an interactive graphics program written in Turbo Pascal for the IBM personal computer. The user types a descriptive command line via the line editor in TRAP. Commands may include lossy dielectrics, lumped elements, and a mirror. The angle of incidence, polarization, wavelength, and layer thicknesses can be varied linearly. The calculated results are displayed as the computations are made. Three real-time keyboard commands are available to stop, speed up, or slow down the simulation. On the average it takes about 30 seconds per layer to complete a plot on an IBM-XT. The programmed optimization routine is based on a multi-dimensional simplex algorithm [12]. It allows the user to fit a model based on transmission-line theory to the measured reflectance and phase of reflection from a multi-layered medium. TRAP calculates the transmittance, absorptance, reflection coefficient of a layered medium by generalizing Berning's algorithm [13] to include the effects due to periodic grids at the interfaces. This algorithm is numerically more efficient than the conventional cascade matrix approach. Equivalent circuit models are derived from physical dimensions of thin screens such as the square grid. One model is based on MacFarlane's quasi-static formula of a strip [7], and the other model is based on a modification of Eisenhart and Khan's theory of a post in a waveguide [9]. In addition the circuit model of a Jerusalem-cross based on Arnaud and Pelow [1] is also available.

The solution for plane wave propagation in a multi-layered medium is a well known boundary value problem in wave analysis. However, in electromagnetic engineering, it is more desirable to make an analogy between the plane wave solutions and the waves along a transmission-line. The electric and magnetic fields are analogous to the voltage and current in a transmission line, and the ratio is called the impedance. The analogy is useful because it allows the designer to take advantage of existing impedance matching techniques in the field of transmission line design and analysis. To exploit this analogy fully, the concept of admittance, the ratio of magnetic field to electric field, is used because the multi-layered media is plane-parallel, which means its equivalent circuit is parallel. Therefore, it is more desirable to use admittance because parallel admittances add. The method of calculation for wave reflection from multi-layered media is discussed next to establish the notation.

Figure 2.5 shows a schematic of a multi-layered medium with thin structures such as a lossy film or a periodic grid at the interface between two layers. The waves are incident from the left. The boundaries are labeled from 0 to N, with 0 being the interface of the incident medium and N being the interface of the final medium. The angle of incidence is θ_o, and the complex refractive index

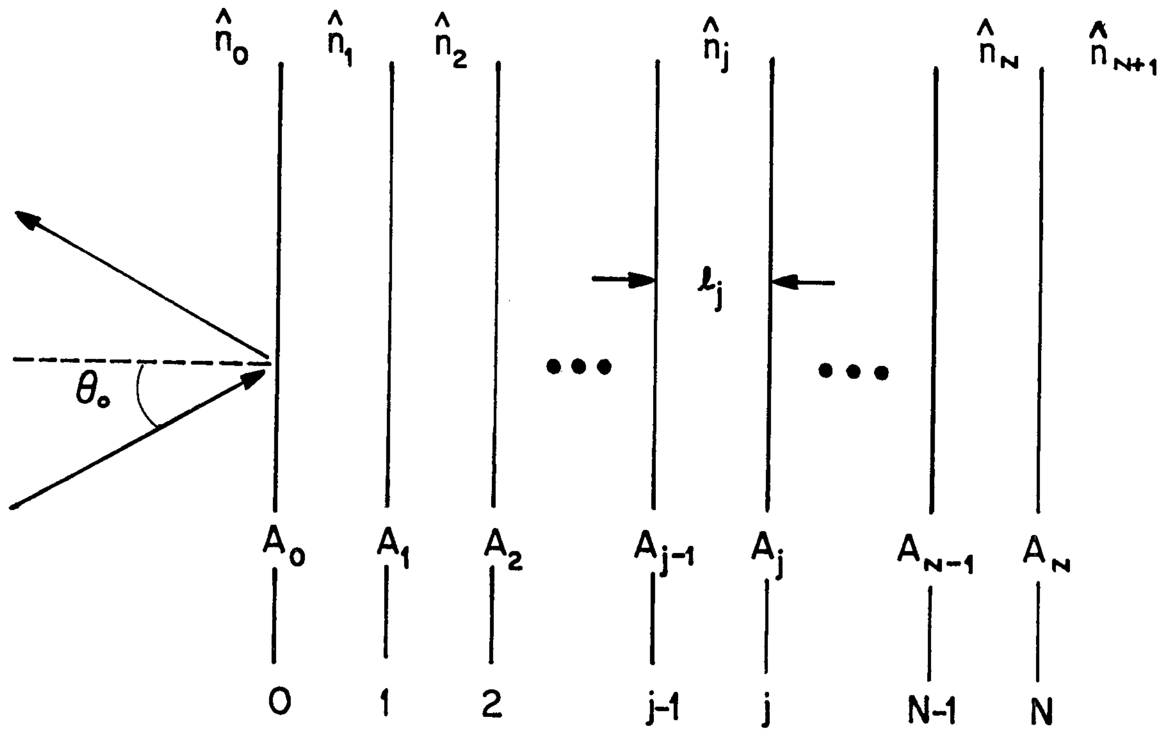


Figure 2.5. Schematic diagram of a multi-layered medium.

of the j^{th} layer is $\hat{n}_j = n_j + ik_j$, where n_j is the refractive index and k_j is the absorption index of the j^{th} layer. The physical thickness of the j^{th} layer is l_j, and its electrical length is $\phi_j = 2\pi\hat{n}_j l_j$. All admittances are normalized to the characteristic admittance in the vacuum. The normalized admittance for a thin structure at the j^{th} interface is y_j. The calculation of the reflection coefficient, transmittance, and absorptance begins from the final medium on the right and works its way back toward the incident medium on the left. The normalized admittance of the final medium is initialized to $Y_N = \hat{n}_{N+1} + y_N$. The normalized admittance looking from the j^{th} interface toward the right is given by the recursive relation,

$$Y_{j-1} = a_{j-1} + ib_{j-1} = \hat{n}_{jp}\frac{Y_j \cos\phi_j + i\hat{n}_{jp}\sin\phi_j}{\hat{n}_{jp}\cos\phi_j + iY_j\sin\phi_j} + y_{j-1}, \qquad (2.4a)$$

where

$$\hat{n}_{jp} = \begin{cases} \hat{n}_j \cos\theta_j & for\ TE\ polaraization \\ \hat{n}_j / \cos\theta_j & for\ TM\ polarization \end{cases}, \qquad (2.4b)$$

and θ_j satisfies Snell's Law of Refraction,

$$\hat{n}_o \sin\theta_o = \hat{n}_j \sin\theta_j. \qquad (2.4c)$$

The reflection coefficient at the input surface of the multi-layered medium is calculated from

$$\rho = \frac{\hat{n}_o - Y_o}{\hat{n}_o + Y_o}. \qquad (2.5)$$

The transmittance at the output surface is given by

$$T = (1 - |\rho|^2)\prod_{j=1}^{N}\psi_j, \qquad (2.6a)$$

where ψ_j is the ratio of time average of the magnitude of Poynting's vector at the j^{th} and $(j-1)^{th}$ interfaces and is given by the formula,

$$\psi_j = \frac{a_j}{a_{j-1}|\cos\phi_j + iY_j \sin\theta_j/\hat{n}_j|^2}. \tag{2.6b}$$

Using the Law of the Conservation of Energy, the total absorptance in the layered medium is given as,

$$A = 1 - T - |\rho|^2. \tag{2.7}$$

2.4 Simulated Performance of Diode-Grid Phase Shifter

Figure 2.6 shows a simulation of the diode-grid phase shifter. The assumed metal parameters are based on the skin-effect formulas for gold. Dielectric properties are taken from Afsar and Button's data [14]. The result indicates that the phase of reflection varies linearly from $-180°$ to $+180°$ as the grid reactance sweeps from a normalized reactance of -1 to $+1$ ($-107\,\Omega$ to $+107\,\Omega$ for the gallium-arsenide substrate). The reflection efficiency varies from a low of 0.49 to a high of 0.57, with an average loss of 2.7 dB. Of this loss, all but a tenth of a dB is due to the series resistance of the diode (assumed to be $10\,\Omega$).

Figure 2.7 shows a family of diode-grid phase shifter performances for normal incidence in air. Gallium-arsenide is assumed for the diode-grid substrate. The grid reactance is assumed to vary from $-107\,\Omega$ to $+107\,\Omega$, and resistance is assumed to vary from $10\,\Omega$ to $50\,\Omega$). The refractive index of the quarter-wavelength dielectric cover is assumed to vary from 2.26 to 2.34. These indices satisfy the condition,

$$2\left(\frac{Z_c}{Z_s}\right)^4 \left(R^2 + Z_s{}^2\right) = Z_i^2, \tag{2.8}$$

where the Z_s is the effective characteristic wave impedance of the diode-grid substrate, Z_c is the effective characteristic wave impedance of the quarter-wave

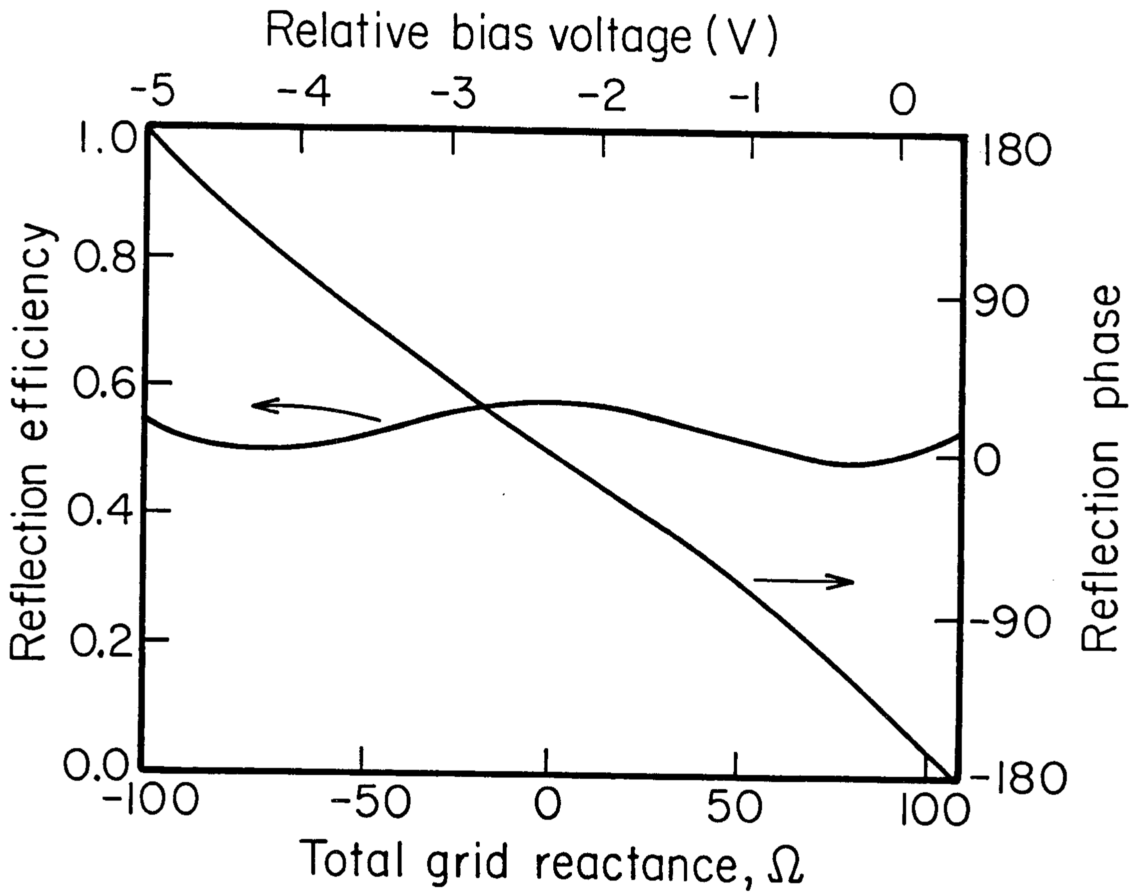


Figure 2.6. Simulated results of the programmable diode-grid phase shifter. The initial quarter-wave transformer layer is fused quartz, the angle of incidence in the air is 45°, and the polarization is *TE*.

dielectric cover, Z_i is the effective characteristic wave impedance of the incident medium, and R is the series resistance of the diode-grid. It is interesting to note that the use of these refractive indices equalizes the reflection losses by having the impedance looking into the diode-grid phase shifter to revolve around the center of the Smith chart. Also, the required refractive indices are slightly larger than the refractive index usually required for an anti-reflection coating. Although the required refractive indices are not too practical from the material point of view, the same effect of equalizing the reflection loss can be obtained by adjusting the angle of incidence in air. When this is done for TE polarized waves, the angle of incidence is about 36° for crystal-quartz, and about 52° for fused quartz. From Figure 2.7 we can see that the maximum and minimum reflection losses occurs around 0° and ±110°, respectively. Figure 2.8 shows these reflection losses as a function of $R/\Delta X$, the ratio of the real part to the total change in the imaginary part of the diode-grid impedance. They vary almost linearly in $R/\Delta X$ from 0.05 to 0.5. The slope for the maximum loss curve is 29.5, and for the minimum loss curve, 24.2.

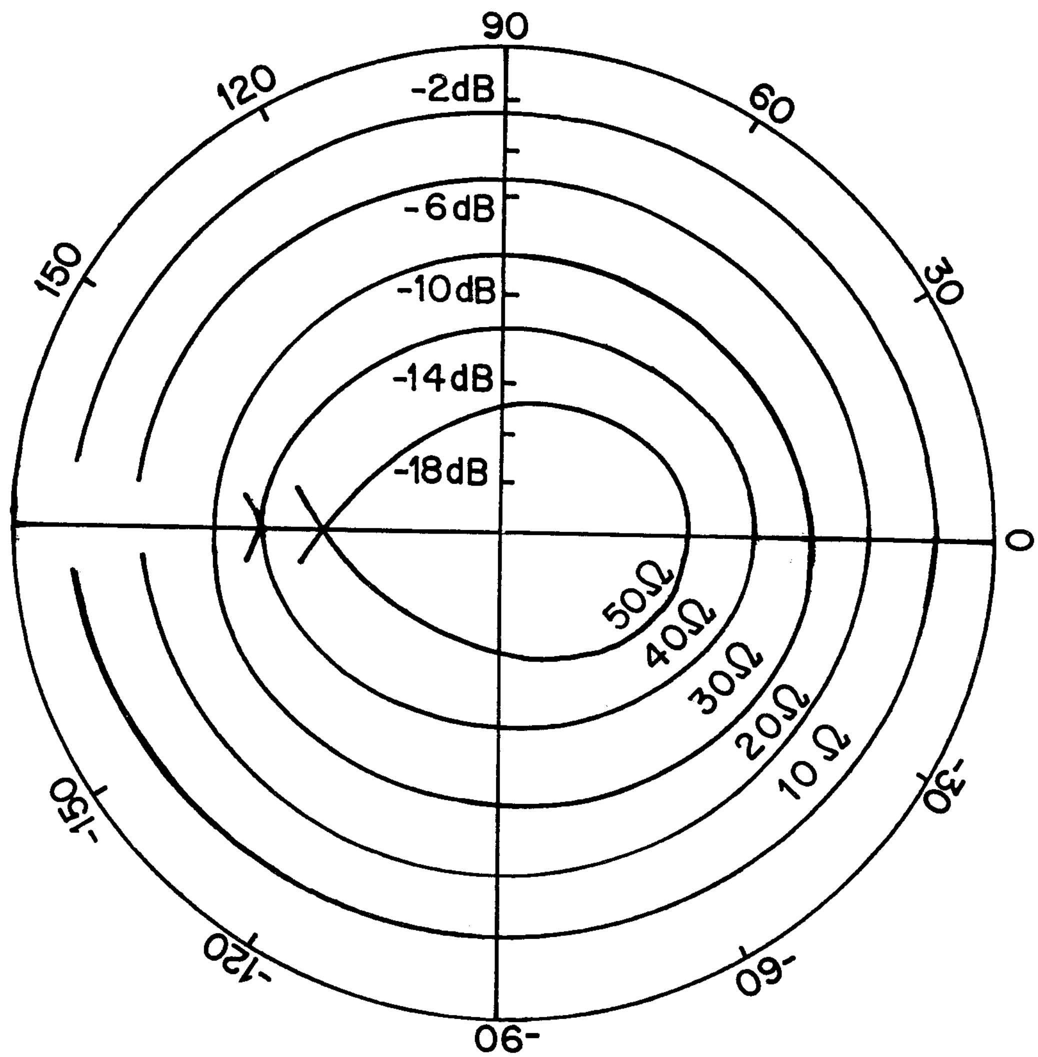


Figure 2.7. Reflection loss of the diode-grid phase shifter

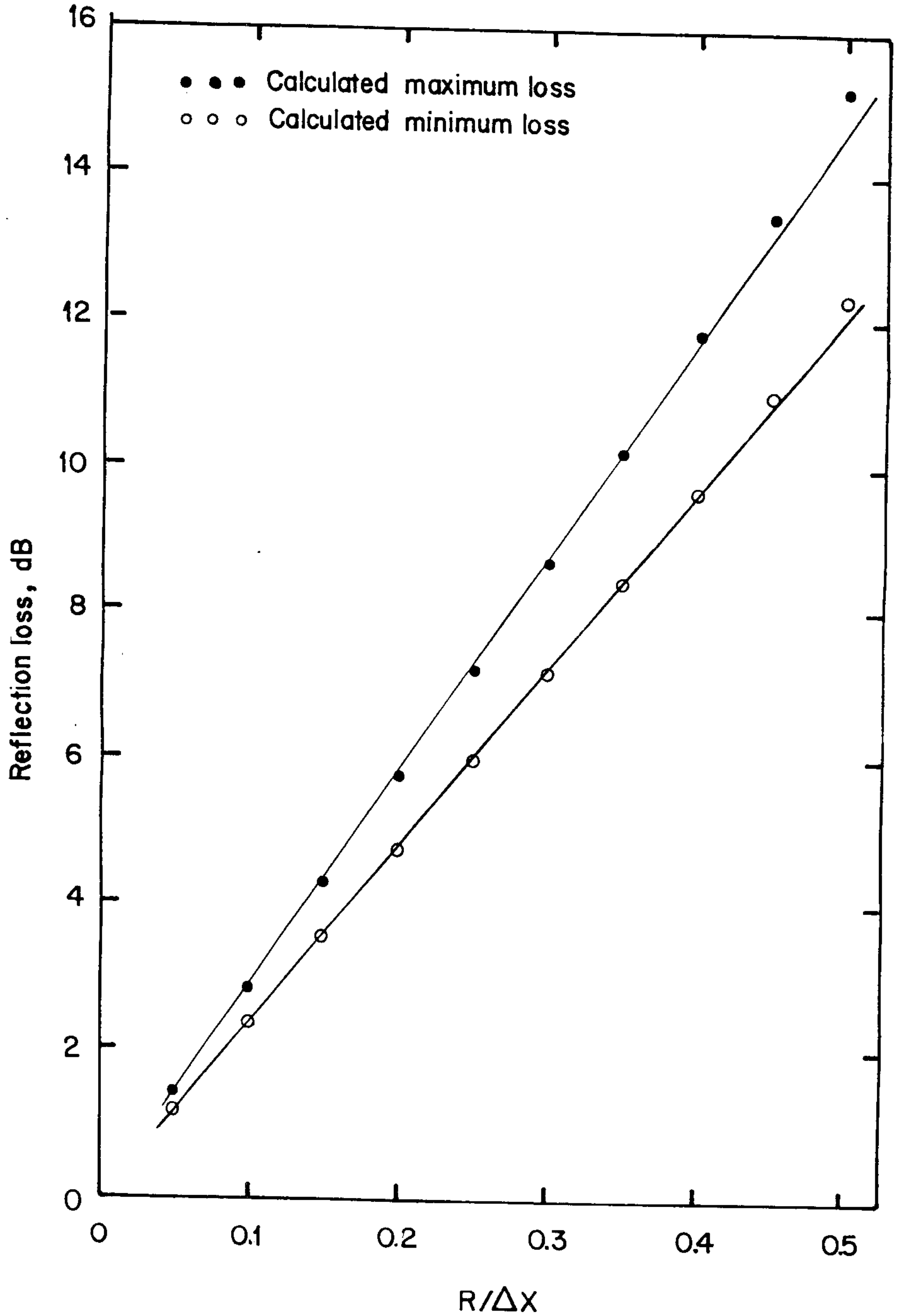


Figure 2.8. Reflectance loss of the diode-grid phase shifter as a function of the ratio of diode-grid series resistance to reactance.

References

[1] J. A. Arnaud and A. Pelow, "Resonant-Grid Quasi-Optical Diplexers," *Bell Sys. Tech. J.*, **Vol. 54**, pp. 263–283, 1975.

[2] R. E. Collin, *Field Theory of Guided Waves*, **Chap. 12**, McGraw-Hill, New York, 1960.

[3] G. Latmiral and A. Sposito, " Radar Corner Reflector for Linear or Circular Polarization," *J. of Res. of NBS*, **Vol. 66D**, pp. 23–29, 1962.

[4] R. Watanabe, "A Novel Polarization Independent Beam Splitter," *IEEE Trans. Mircrowave Theory and Tech.*, **MTT-28**, pp. 685–689, 1980.

[5] N. Nakajima and R. Watanabe, "A Quasi-Optical Circuit Circuit Technology for Shortmillimeter-Wavelength Multiplexers," *IEEE Trans. Microwave and Tech.*, **MTT-28**, pp. 897–905, 1981.

[6] R. Ulrich, T. J. Bridges, and M. A. Pollack, "Variable Metal Mesh Coupler for Far Infrared Lasers," *Appl. Opt.*, **Vol. 11**, pp. 2511–2516, 1970.

[7] G. G. MacFarlane, "Quasi-Stationary Field Theory and Its Application to Diaphragms and Junctions in Transmission Lines and Wave Guides," *Proc. Inst. Elect. Eng.*, **93, Part 3A**, pp. 1523–1527, 1946.

[8] J. R. Wait, "Impedance of a Wire Grid Parallel to a Dielectric Interface," *IRE Trans. on Microwave Theory and Tech.*, **MTT-5**, pp.99–102, 1957.

[9] D. B. Rutledge and S. E. Schwarz, "Planar Multimode Detector Arrays for Infrared and Millimeter-Wave Applications," *IEEE J. of Quantum Electronics* **QE-17**, pp. 407-414, 1981.

[10] R. V. Garver, "360° Varactor Linear Phase Modulator," *IEEE Trans. on Microwave and Tech.*, **MTT-17**, pp. 137–147, 1969.

[11] A. Y. Cho and F. K. Reinhart, "Interface and Doping Profile Characteristics with Molecular-Beam Epitaxy of GaAs: GaAs Voltage Varactor," J. of

Applied Physics, **Vol. 45**, pp. 1812–1817, 1974.

[12] J. A. Berning and P. H. Berning, "Thin Film Calculations using the IBM 650 Electronic Calculator," *J. of the Opt. Soc. of America*, **Vol. 50**, pp. 813–815, 1960.

[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, **Chap. 10**, pp. 274–334, Cambridge University Press, New York, 1986.

[14] M. N. Afsar and K. J. Button, "Precise Millimeter-Wave Measurements for Complex Refractive Index, Complex Dielectric Permittivity and Loss Tangent of $GaAs$, Si, SiO_2, Al_2O_3, BeO, Macor, and Glass," *IEEE Trans. on Microwave Theory Tech.*, **MTT-31**, pp. 217–223, 1983.

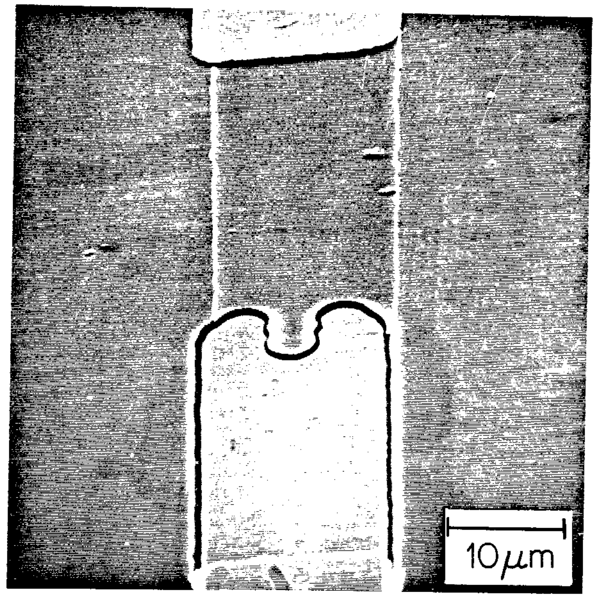
Chapter 3

Fabrication of Diode-Grids

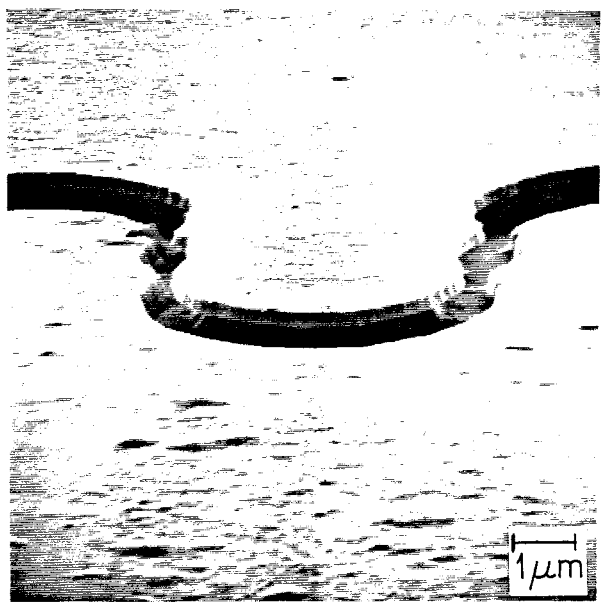
Monolithic diode-grids have been fabricated on 2 cm by 3 cm gallium-arsenide (GaAs) wafers with 2000 aluminum Schottky-barrier varactor diodes. The self-aligning technique, which Zah had developed in our group [1], is used to fabricate the diodes. The best fabrication yield for individual diodes in an array was 98 %. A liquid crystal detection technique was developed to identify the diodes that are shorted in the grid, and subsequently an ultrasonic probe is used to remove the bad diodes. The design of the varactor diode with a truncated-hyperabrupt doping profile is discussed. The doping profile is measured with a mercury probe. The diode series resistance is calculated from the current-voltage (IV) characteristics measured at DC, and the capacitance parameters are calculated from the capacitance-voltage (CV) characteristics measured at 1 MHz. The detailed procedures are given in the appendix.

3.1 Design of a Hyperabrupt Schottky Varactor Diode

Figure 3.1 shows SEM photographs of the Schottky-barrier varactor diode on GaAs. The diode consists of a Schottky contact with a shape of a strip, which is surrounded by an ohmic contact. The width of the strip is defined by a self-aligning technique, and its length is defined by proton bombardment. This is the key of Zah's process, which enables us to make a small planar diode for millimeter-wave applications. The area of the strip is about 18 μm^2; its small rectangular geometry gives a low spreading resistance since the periphery-to-area ratio is high. The metal for the Schottky contact is aluminum (Al), and the metal for the ohmic contact is gold-germanium/nickel/gold (AuGe/Ni/Au). The metalization extends from the diode and becomes part of inductive lead of the



(a)



(b)

Figure 3.1. SEM photographs of a planar Schottky-barrier varactor diode. (a) Diode is located at the tip of the strip. (b) Close-up view of diode.

diode-grid. The other part of the inductive lead is gold. For a large capacitance variation, the varactor is designed with a hyperabrupt doping profile; that is, the net doping concentration of the epitaxy decreases with the distance from the metal semiconductor interface.

The three most important circuit parameters of a varactor are the breakdown voltage V_b, the zero bias capacitance C_o, and the series resistance R_s. They affect the amount of power the varactor will handle, the level of impedance the varactor will present, and the amount of power the varactor will dissipate in a circuit. These parameters can be calculated and optimized when the doping concentration as a function of the distance from the junction is specified. Norwood and Shatz [2] analyzed an ideal hyperabrupt doping profile that is described mathematically by an mth power law. In practice, a truncated-hyperabrupt doping profile must be used. Figure 3.2 shows the truncated-hyperabrupt doping profile. The doping concentration is given by

$$N_d = \begin{cases} N_\circ & 0 \leq x \leq x_\circ \\ N_\circ\left(\frac{x}{x_\circ}\right)^m & x_\circ \leq x \leq T_{epi} \end{cases}, \tag{3.1}$$

where x_o is the zero bias depletion width, N_o is the doping concentration at the surface, T_{epi} is the epitaxial thickness, and m is the doping profile exponent. The method of design is based on Lundien *et al.*'s [3] design algorithm of an exponentially retrograded doping profile. In the *depletion approximation*, the one-dimensional Poisson equation,

$$\frac{d^2\phi}{dx^2} = -\frac{q}{\epsilon_s}N_d, \tag{3.2}$$

is integrated, where ϕ is the electric potential, q is the electronic charge, and ϵ_s is the dielectric permittivity of the semiconductor. The boundary conditions are

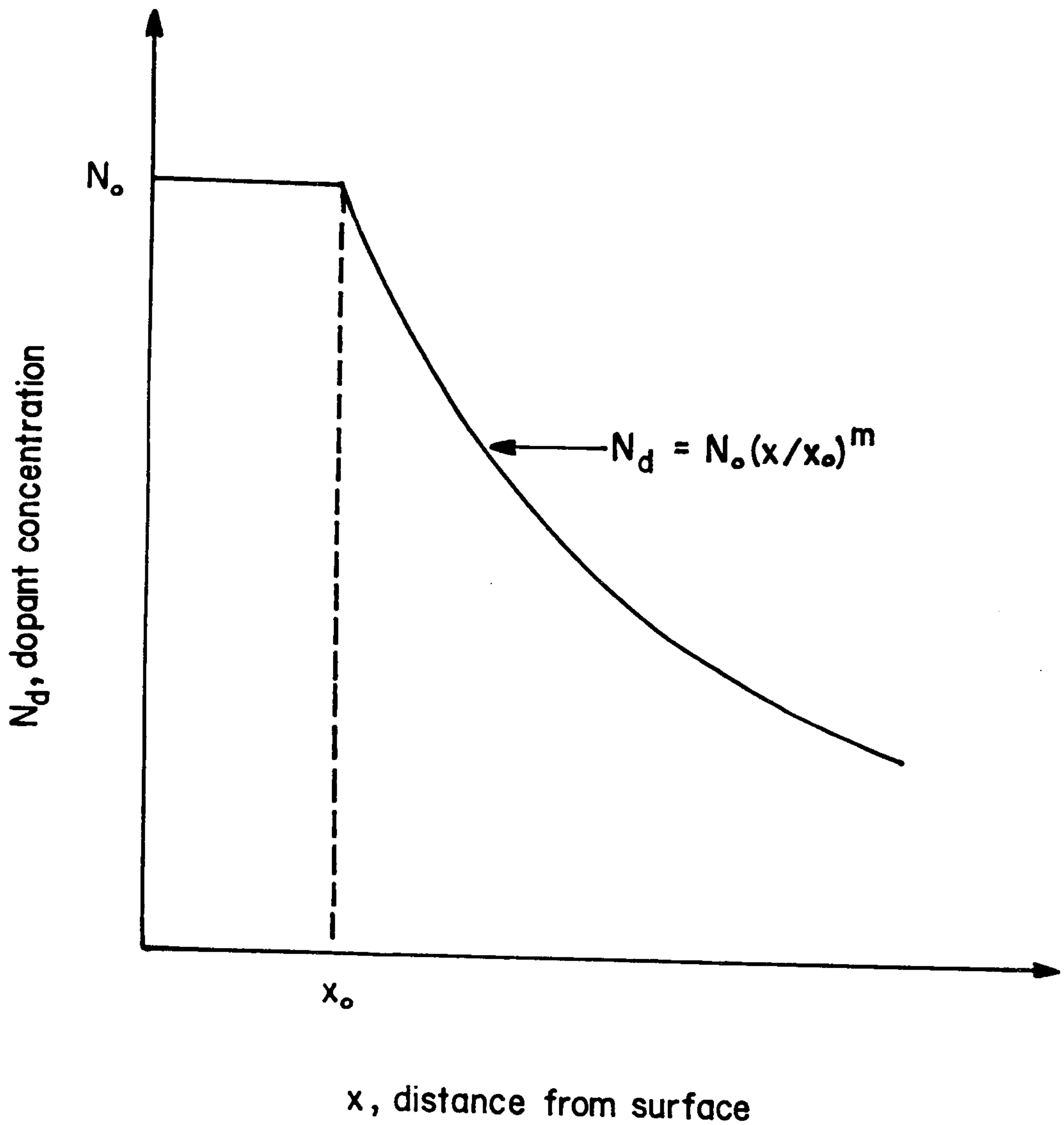


Figure 3.2. A truncated-hyperabrupt impurity doping distribution designed for a varactor diode.

$\phi(x=0)=0$ and $\phi(x=W_{dep})=\phi_j-V$, where ϕ_j is the junction potential, W_{dep} is the depletion width of the space-charge region, and V is the applied voltage. This leads to a complicated algebraic relation, $V=F(W_{dep})$, which is used to compute the corresponding CV relation

$$C=\epsilon_s\frac{A}{W_{dep}}, \tag{3.3}$$

where A is area of the varactor. The calculated CV characteristic is checked for self-consistency by the following expression [4],

$$N_d=\frac{\left(\dfrac{-C^3}{q\epsilon_s A^2}\right)}{\left(\dfrac{dC}{dV}\right)}. \tag{3.4}$$

The profile computed from the above is compared with the original profile. The resistance due to the undepleted epitaxial layer is calculated from,

$$R_{epi}=\int_{W_{dep}}^{T_{epi}}\frac{dx}{q\mu_n N_d A}, \tag{3.5}$$

where μ_n is the electron mobility, and T_{epi} is the epitaxial thickness. An emperical expression for the electron mobility is used

$$\mu_n=\frac{10^4}{1+\sqrt{\dfrac{N_d}{10^{17}}}}, \tag{3.6}$$

where N_d is in units of cm^{-3} and μ_n is in units of $cm^2(Vs)^{-1}$ [5]. The breakdown voltage is computed as the applied voltage at which the ionization integral becomes unity,

$$I=\int_0^{W_{epi}}A\exp\left[-\left(\frac{b}{E(x)}\right)^2\right]dx=1, \tag{3.7}$$

where $E(x)$ is the electric field in units of $\mathrm{V(cm)^{-1}}$, $A = 3.5 \times 10^5\ \mathrm{cm^{-1}}$, and $b = 6.85 \times 10^5\ \mathrm{V(cm)^{-1}}$ [6]. The integrals are integrated numerically with an algorithm based on Simpson's rule [7]. The electric field is obtained from the one-dimensional Poisson equation. The boundary conditions are imposed by matching the fields at $x = x_\circ$ and by setting the field to zero at $x = W_{epi}$. The expressions obtained are checked with those obtained from Gauss's law.

A simple graphical procedure is used as a guide to design the truncated-hyperabrupt profile for a varactor. The design parameters are the surface concentration ($N_\circ$), the doping profile exponent (m), the zero bias depletion width ($x_\circ$), and the epitaxial thickness. A figure of merit for a varactor is the dynamic cutoff frequency, which is defined by Penfield and Rafus [8] to be,

$$f_c = \frac{S_{max} - S_{min}}{2\pi R_s}, \tag{3.8}$$

where S_{max} is the reciprocal of minimum capacitance, S_{min} is the reciprocal of the maximum capacitance, and R_s is the series resistance of the varactor. Other quantities including series resistance, capacitance tuning ratio, and avalanche integral are also calculated to indicate design margins and tradeoffs.

Figure 3.3 shows a contour plot of dynamic cutoff frequency as a function of surface concentration and doping profile exponent. An approximated junction potential for Al on GaAs is 0.94 V from Eglash *et al.* [9]. The maximum capacitance is taken to be the zero bias capacitance, which is assumed to be 30 fF because this gives the desired reactance at 90 GHz for our experiments. The epitaxial thickness is 0.65 μm. This thickness is chosen because it is thick enough to give a capacitance ratio of 5 and thin enough for proton isolation. The calculation also assumes a parasitic resistance of 7 Ω and a parasitic capacitance of 3 fF. The solid line indicates that the ionization integral is unity ($I = 1$) and divides

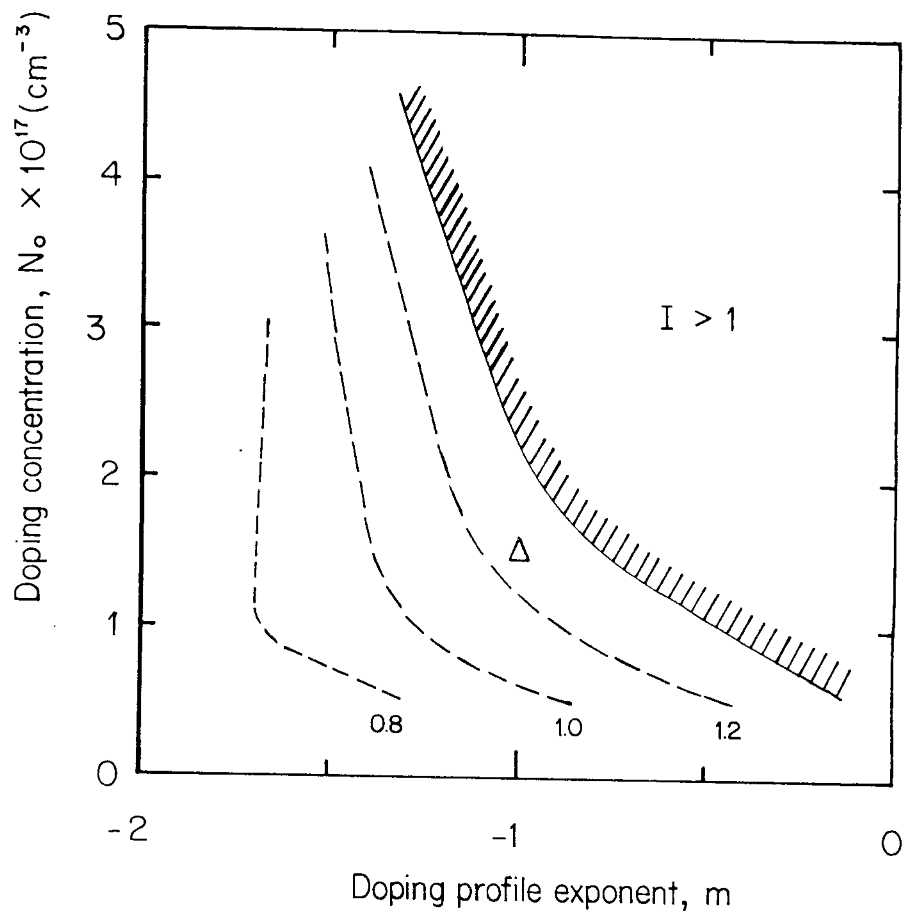


Figure 3.3. A contour plot of dynamic cutoff frequency in units of THz as a function of surface concentration and doping profile exponent in the region where the ionization integral is less than unity $(I < 1)$. The solid line corresponds to the $(I = 1)$ and the shaded region corresponds to the $(I > 1)$.

the space into safe and unsafe regions of operation. Safe means that the entire epitaxial layer can be depleted without avalanche breakdown or with ($I < 1$). Qualitatively speaking, the closer N_o and m approach the solid line, the greater the chance of reaching avalanche breakdown. The bigger N_o is, the bigger the capacitance ratio, which is good for phase-shifting, while ($-0.5 < m < -0.3$) is more favorable for second harmonic conversion efficiency [10]. The final design of the truncated-hyperabrupt doping profile varactor is $N_o = 1.5 \times 10^{17}$ cm^{-3}, $m = 1.0$, $x_o = 0.1\,\mu$m, and $T_{epi} = 0.65\,\mu$m. An extra 0.05 μm is added as safety margin for back depletion from the n$^+$ layer. The thickness of the n$^+$ layer is 1.8 μm. This gives a total epitaxial thickness of 2.5 μm, which is close to the limit of the proton isolation capability available to us. The doping concentration for the n$^+$ layer is designed to be 3×10^{18} cm^{-3}.

A mercury probe is used to measure the CV characteristic and the doping profile [10]. Figure 3.4 shows the measured CV characteristic. The ratio of the capacitance at zero bias to the capacitance at breakdown is about 4.5 and this is close to what the simulation predicts. The breakdown voltage is about 9 V, and this corresponds to what the simulation predicts when the ionization integral is about 0.4. This comparison is based on averaging the calculated results at $m = -1.1$ and $m = -0.9$. Figure 3.5 shows the corresponding measured doping profile and compares it to the profiles that were designed and measured with a *Polaron* profiler. Basically, a back-to-back Schottky diode is formed on the surface of the wafer with a small and a large mercury dot, which are held there by a vacuum. Then the small signal capacitance of this diode structure is measured as a function of the DC bias. The advantage of this technique is that it is non-destructive. Occasionally, the doping profile is available from a *Polaron* profiler for comparison. This provides a more complete measure because it includes the n$^+$ layer; however, this is a destructive technique since it requires etching away

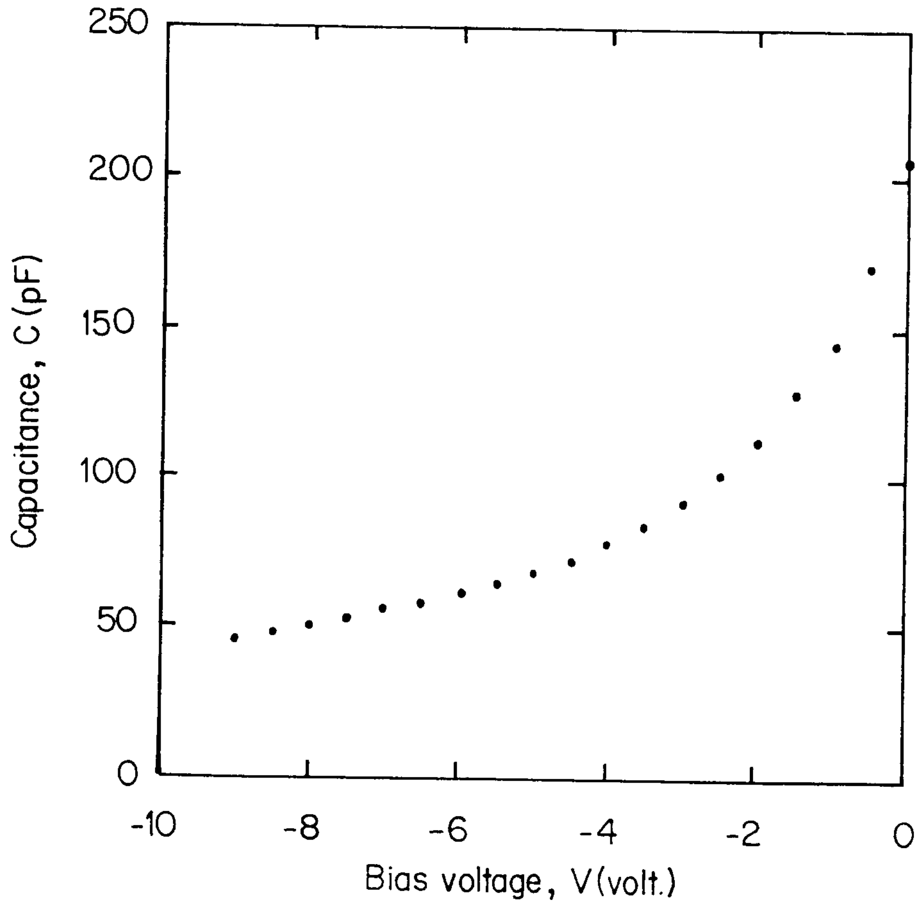


Figure 3.4. Measured capacitance-voltage characteristic at 1 MHz from a GaAs wafer with a truncated-hyperabrupt doping profile.

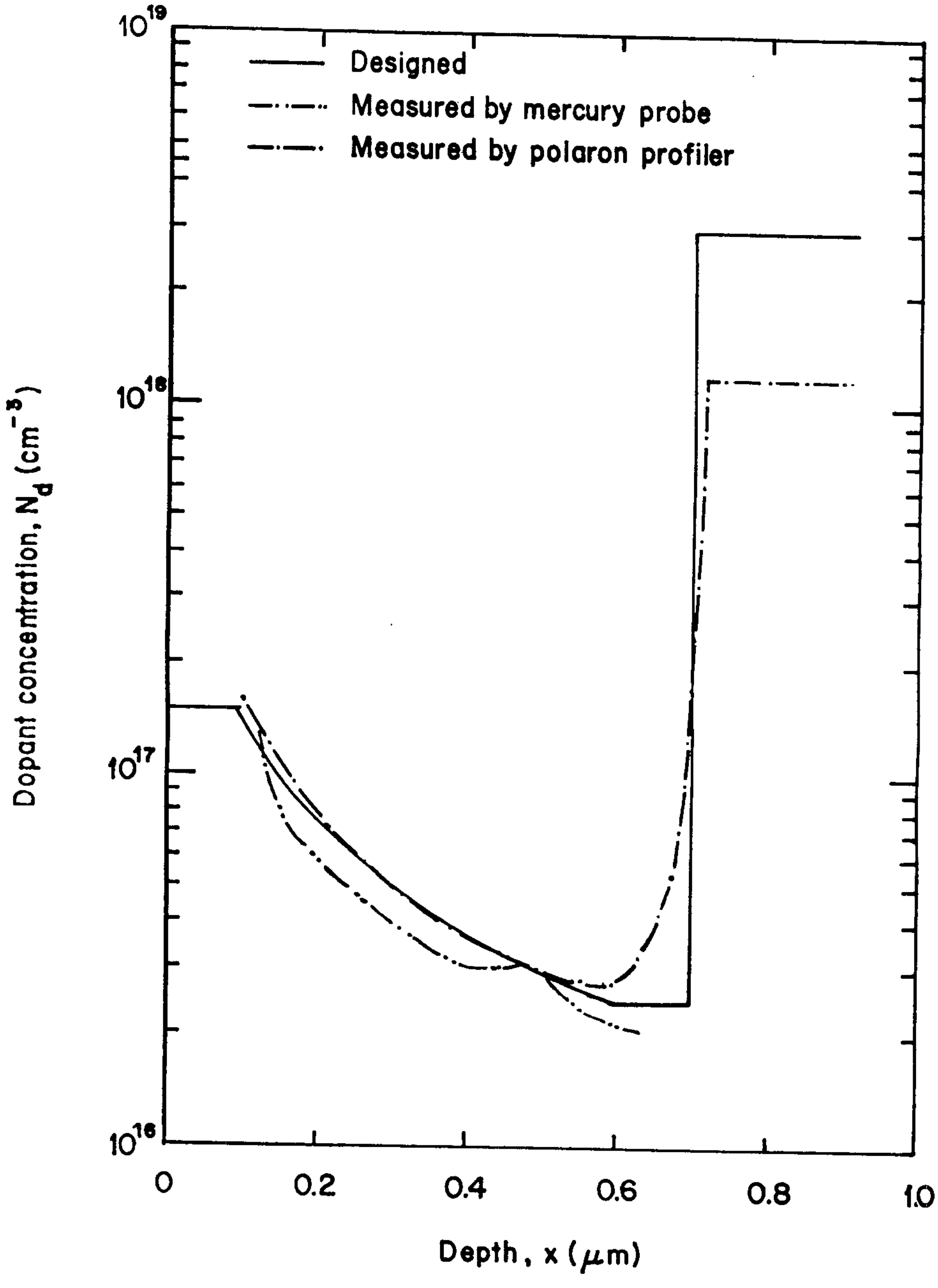


Figure 3.5. A comparison of designed doping profile with the measured doping profiles from a mercury probe and a *Polaron* profiler.

the substrate in determining the depth. The doping profile is calculated from the measured CV characteristic according to Equation (4). This is run by an IBM-PC that controls an HP4280A C-meter for CV measurements. The CV program is given in the appendix.

3.2 Fabrication Processes

Having determined the CV characteristic and doping profile of the wafer, we evporated a 2000 Å thick layer of aluminum. Although *in-situ* molecular beam epitaxy (MBE) aluminum is superior [1,12], it is not as readily available. Figure 3.6 summarizes all the different layers of the starting material for fabricating the diode-grid. The aluminum is on an 0.7 μm layer of n-type GaAs with a hyperabrupt doping profile. The n-type GaAs is on an 1.8 μm layer of n^+ GaAs with a doping concentration of $1 \times 10^{18}\,\mathrm{cm}^{-3}$. Howard Chen, of Professor Yariv's group at Caltech, and Kjell Stolt of TRW grew the epitaxial layers on chrome-doped semi-insulating GaAs with MBE for us. Wafers as large as 2 cm by 3 cm have been used to fabricate the diode-grids. The diodes are fabricated with a self-aligning process that Zah developed [1].

3.2.1 Self-Aligning Schottky Contact

Figure 3.7 shows how the self-aligning process works. A photoresist is patterned to protect the aluminum Schottky contact during wet etching and to serve as a lift-off mask when the metalization for the ohmic contact is evaporated. The structure is formed by first etching the aluminum until it cuts under the photoresist and then etching the n-layer until it reaches the n^+ layer and cuts under the photoresist. When AuGe/Ni/Au is evaporated over this structure, a small gap is created between the Schottky metal and the ohmic metal; hence, the Schottky contact is self-aligned to the ohmic contact.

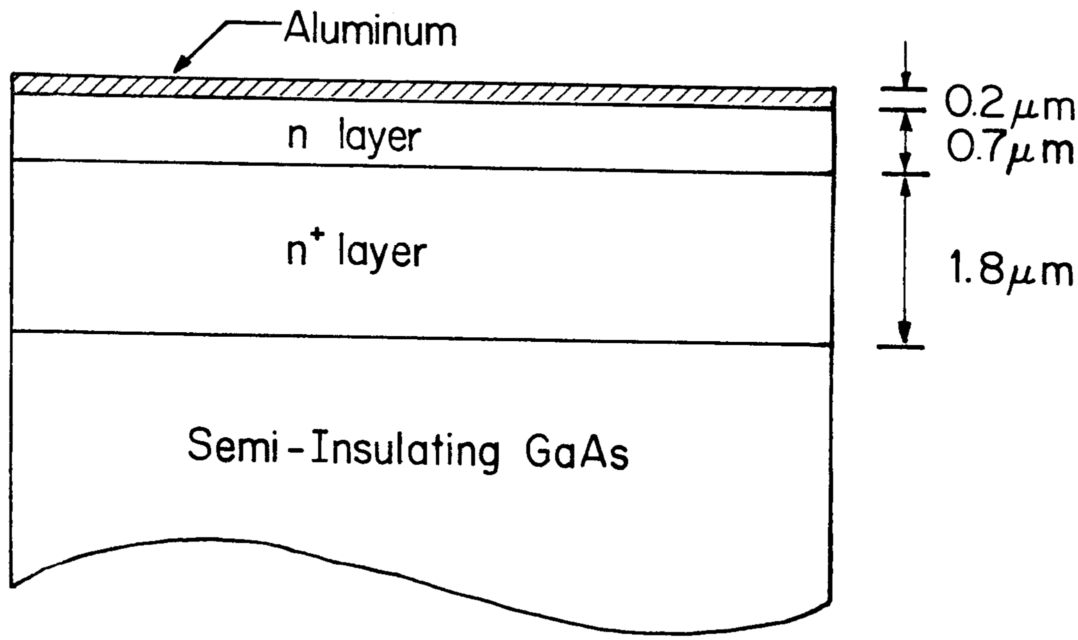


Figure 3.6 A side view of the starting material with 2000 Å of evaporated aluminum on the n-type GaAs epitaxy.

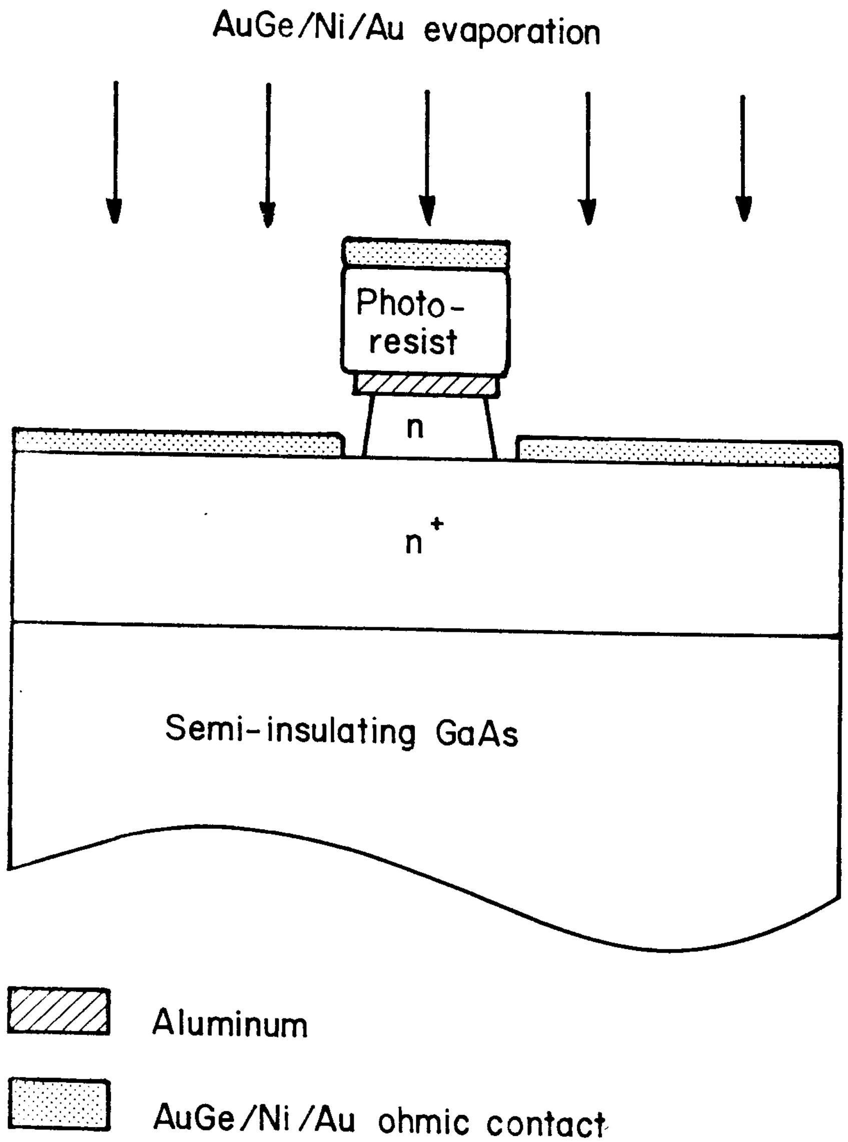
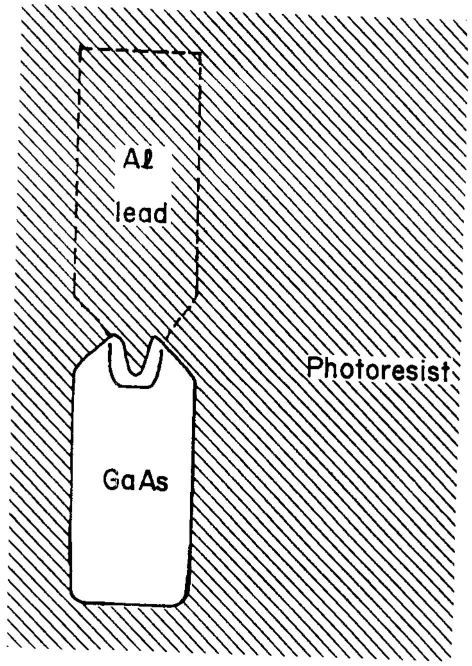


Figure 3.7. Zah's self-aligning process for defining the width of the Schottky contact.

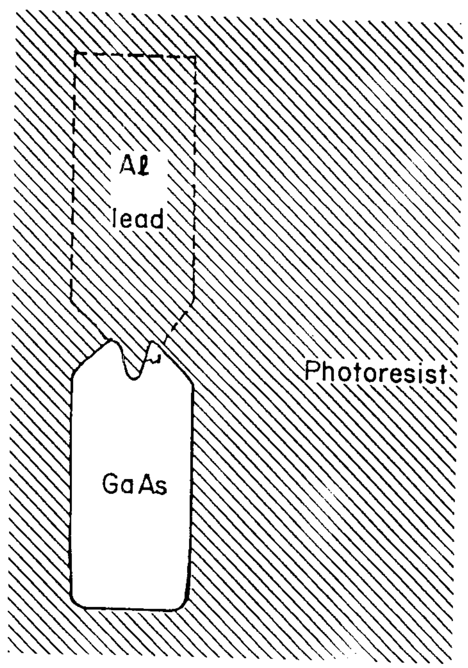
In the past, the lead for the Schottky contact was patterned first and then the width of Schottky contact and the ohmic contact were defined by the self-aligning process as described above. Also *in-situ* MBE aluminum was used. In my experience of using evaporated aluminum, I find it better to etch the width of the Schottky contact and the ohmic contact first and then pattern the lead for the Schottky contact. This has the following advantages. Since a small amount of aluminum is removed in defining the contacts, there is still a significant amount of aluminum left over. If over-etching occurs, one can repeat this step several times until the desired result is obtained. More importantly, etching becomes easier to control because etching tended to be less uniform, when the lead for the Schottky contact was etched before the width of the Schottky contact and the ohmic contact were etched. This is probably due to bubbles formed and trapped at the corner of the photoresist pattern. Figure 3.8 illustrates this problem. Having patterned the lead for the Schottky contact, we then developed the photoresist mask for self-aligning the diode contacts via etching. This exposes a small piece of aluminum to be etched away. Figure 3.8a shows the top view of this. When the wafer is dipped into aluminum etchant, it was found that about 20 % of the diodes tended to be incompletely etched. Typically a small patch of aluminum is left behind as shown in figure 3.8b. If the wafer is etched again, then those diodes that were etched completely will be over-etched. If the wafer is not etched again, then those diodes with a small patch of aluminum cause an electrical short when the ohmic metalization is evaporated. This problem is corrected by simply reversing these two steps.

3.2.2 Ohmic Contact

The technique of using AuGe/Ni/Au to form an ohmic contact on n-type GaAs in a furnace is followed [13]. The thicknesses for the metalizations are 700 Å



(a)



(b)

Figure 3.8. Schematic for illustrating a fabrication problem. (a) Before etching the width of the diode. (b) After etching the width of the diode, which has a small patch of aluminum due to nonuniform etching.

of AuGe (88 % Au and 12 % Ge by weight), 300 Å of Ni and 2000 Å of Au. They are evaporated consecutively without breaking the vacuum. The metalizations for the bonding pads are also evaporated at the same time. This is because the alloy process makes the bonding pads adhere to the substrate better and roughens the surface of bonding pads so that they are easier for ultrasonic wire bonding. The alloying process is done in a furnace at 460 °C for 10 minutes with flowing forming-gas. As the AuGe alloy begins to melt, gallium diffuses into the metal [14]. Germanium diffuses into the crystal lattice and dopes the GaAs. Nickel enhances this diffusion and keeps the metal 'wet' onto the surface from segregating in lumps, and gold serves as a capping layer to increase the conductivity.

Making a good ohmic contact is important in achieving a low series resistance for the varactor diode. Factors that influence the quality of an ohmic contact are well documented [14,15]. One of the most important factor that influences the quality of an ohmic contact is the alloying temperature. Other parameters including the type of metalization and its composition and thickness, ambient gas, alloying time etc. can easily be reproduced based on published literature. There is a wide variation in the temperature used by various laboratories, because the equipment and the way in which alloy temperature is measured are not the same; therefore, it is necessary to calibrate the temperature of our furnace controller. The basic technique of using a linear resistor array to measure the contact resistance is used [14]. Figure 3.9 is a photograph of the actual array fabricated on a GaAs wafer. The ohmic contacts are AuGe/Ni/Au alloyed on a mesa of n^+ GaAs epitaxy. An HP3478A multimeter with a 4-wire measurement capability is used to measure the resistance between the ohmic contacts. The measured resistance as a function of distance gives the contact resistivity. The furnace controller is calibrated by measuring the contact resistivity of the

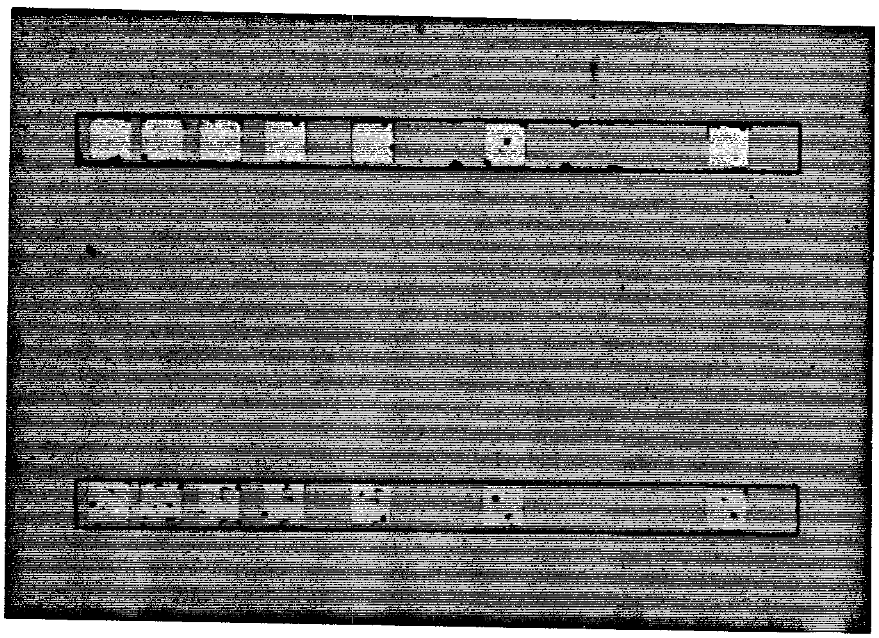


Figure 3.9. Photograph of the resistor array used in measuring contact resistance. Square ohmic contacts are separated by increasing distances. Each resistor array is isolated by mesa etch.

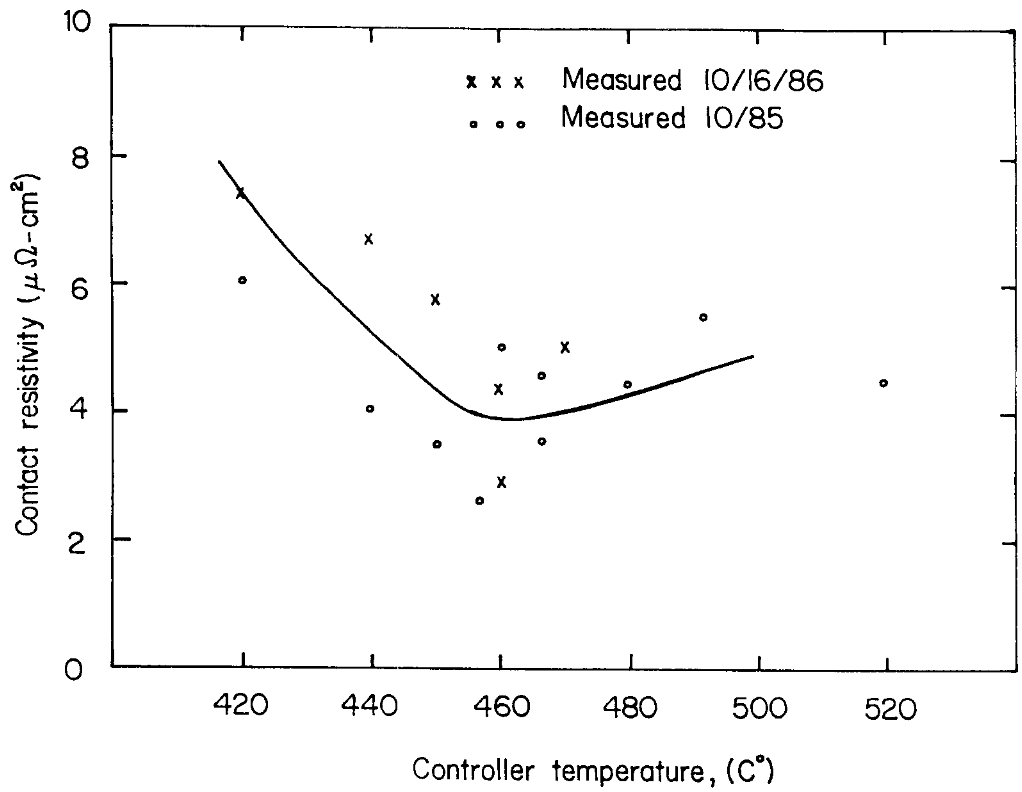


Figure 3.10. Temperature calibration curve of the furnace controller used in alloying ohmic contacts.

ohmic contacts alloyed at various temperatures. Figure 3.10 shows the temperature calibration curve for our furnace controller. A minimum contact resistivity of $4\,\mu\Omega(\mathrm{cm})^2$ was measured at an alloying temperature of 460 °C. The actual temperature was estimated to be 430 °C.

3.2.3 Proton Isolation

Figure 3.11 shows the proton isolation process used in defining the length of the diode. Figure 3.12 shows a $7\,\mu$m thick photoresist patterned to protect the diode from the protons. The implanted protons convert the n-type semiconductor into a high resistivity dielectric by creating deep levels that trap free carriers [15]. Two consecutive proton bombardments are used in order to completely isolate the epitaxial layers. The implantation parameters are 1.) dose=4×10^{14} cm^{-2} and energy=330 keV, and 2.) dose=4×10^{14} cm^{-2} and energy=200 keV. This was done for us by Frank So and Ali Ghaffari in Dr. Nicolet's group at Caltech and Bob Rush at Hudges. The temperature reached during implantation is high enough to harden the photoresist, so an oxygen plasma is used to remove the photoresist.

3.2.4 Low Frequency Varactor Parameters

After proton isolation, low frequency parameters of the varactor are measured. A number of varactors are sampled through out the wafer in order to assess the amount of nonuniformity, and to find an average and a standard deviation for the nonuniformity. Figure 3.13 shows a contour plot of measured series resistance as a function of position on the wafer. The measured series resistance is based on an algorithm that curve-fits the measured IV characteristic with the following equation [1]

$$I = I_s exp[(V - IR_s)/nV_T] - 1, \qquad (3.9)$$

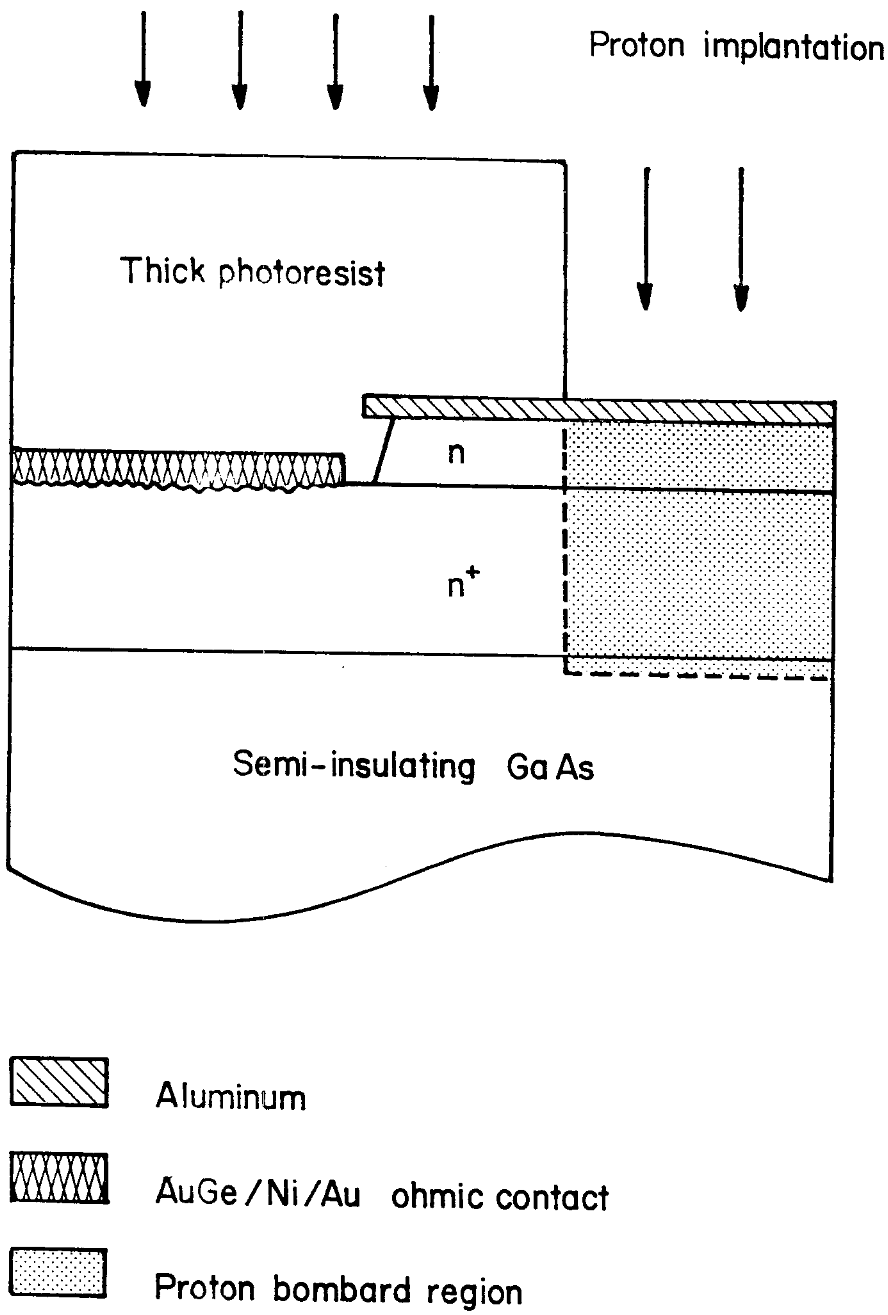


Figure 3.11. Schematic diagram of proton bombardment for defining the length of the Schottky contact.

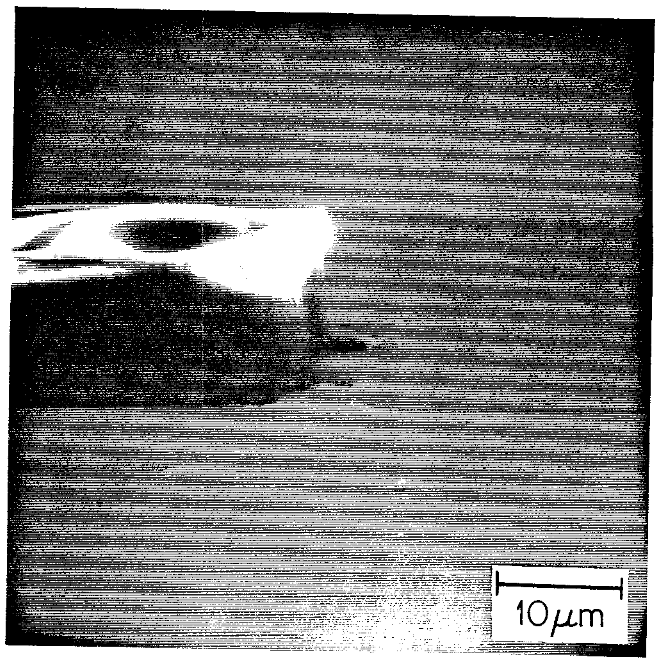


Figure 3.12. SEM photograph of a 7 μm thick photoresist mask for protecting the Schottky contact in defining its length.

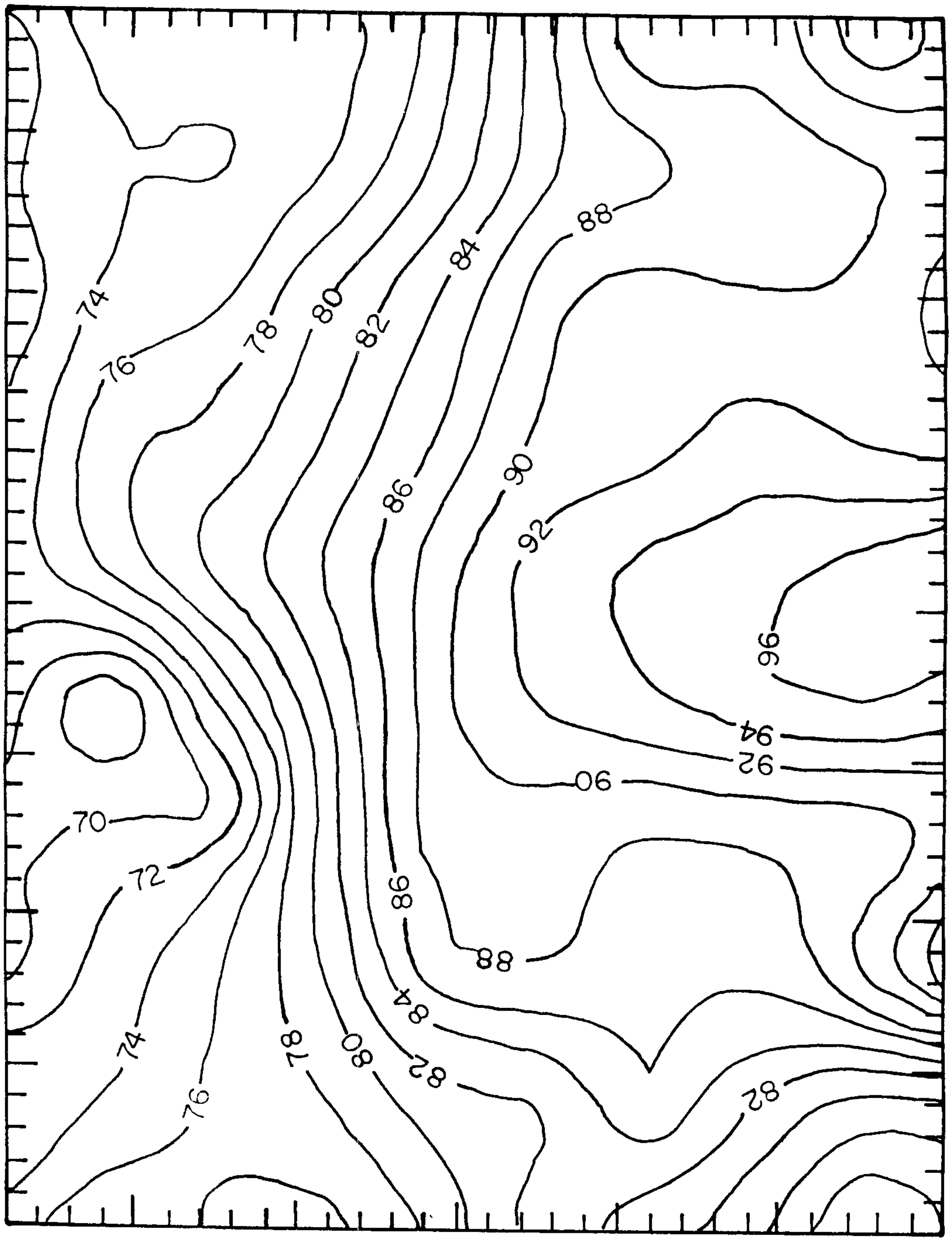


Figure 3.13. A contour map of the diode series resistance in units of ohms. Tick marks correspond to diode positions on the wafer.

where I_s is the reverse saturation current, R_s is the series resistance, n is the ideality factor, q is the electronic charge, k is the Boltzman constant, and T is the temperature. Figure 3.14 shows a contour plot of zero bias capacitance measured at 1 MHz as a function of position for the same wafer. The diode parameters vary considerably. This is mainly due to mask variation and misalignment during fabrication. The measured diode series resistance is 78 Ω with a standard deviation of 19 Ω, zero bias capacitance is 30 fF with a standard deviation of 10 fF, and breakdown voltage is 5.1 V with a standard deviation of 1.9 V. This is based on sampling 95 out of 2000 possible diodes. The diodes have a soft breakdown characteristic. This is probably due to contamination because aluminum is evaporated in a oil diffusion-pumped system at 3×10^{-6} Torr. Similar observations were reported in the literature [9,17,18]. Attempts to use refractory metals such as titanium and molybdenum were made, but no significant improvements have been obtained. The series resistance is quite high because the n^+ concentration is only about 1×10^{18} cm^{-3}. The low breakdown voltage limits the capacitance variation from 14.5 fF at −5.1 V to 52.1 fF at +0.4 V. This corresponds to a capacitance ratio of 3.7. These measurements are computerized. An IBM personal computer is used to control an HP4145B semiconductor parameter analyzer to make the IV measurement and an HP4280A C-meter to make the CV measurement. Software documentations are given in the appendix.

3.2.5 Liquid Crystal Detection

A layer of 2000 Å thick gold and a layer of 100 Å thick chrome are evaporated to define the periodic grid that connects the varactors row by row. The chrome acts as an adhesion layer between gold and GaAs. Despite the fact that the yield of the number of devices with a diode characteristic is quite high, the remaining bad diodes, which tend to be electrical shorts, render the entire diode-

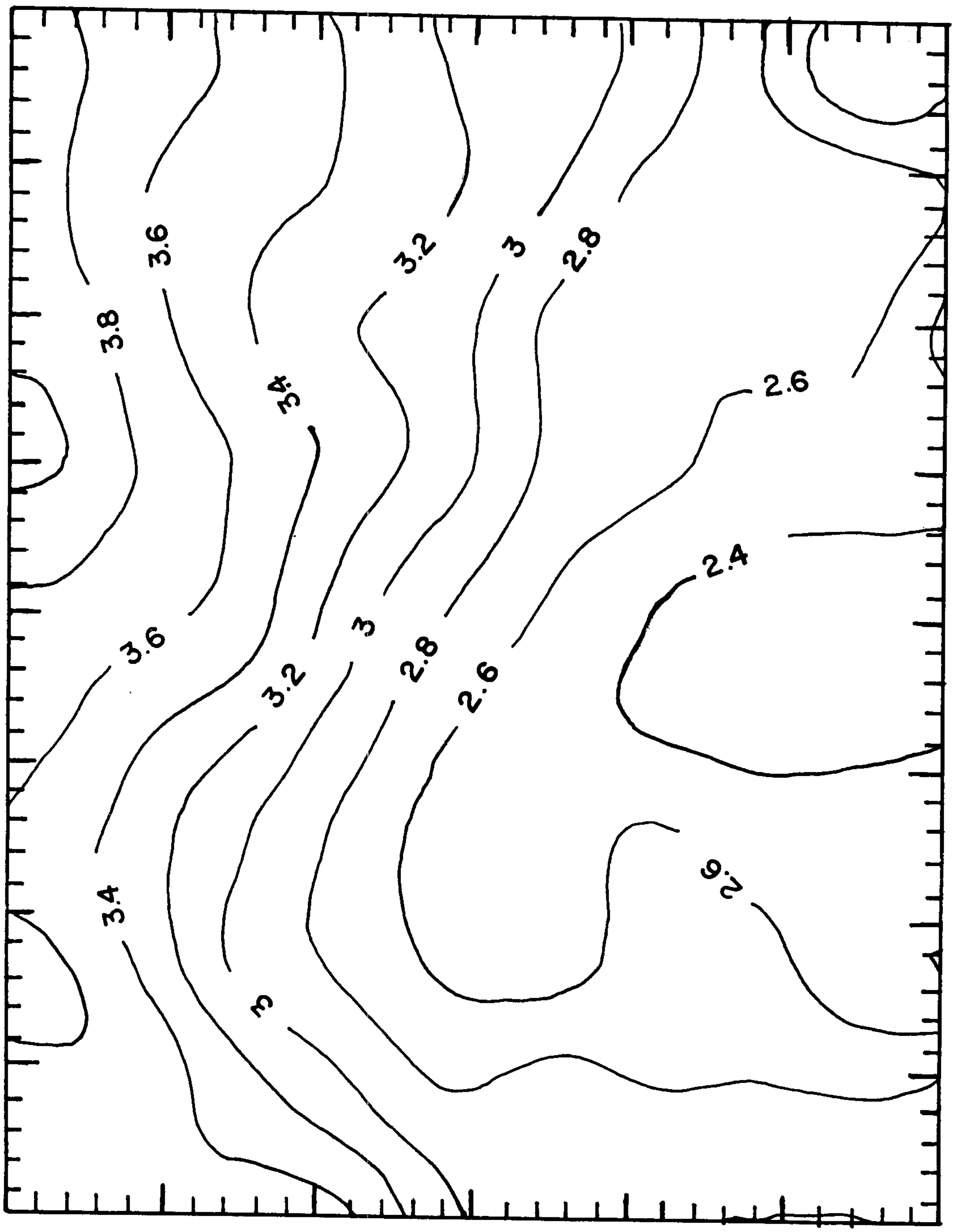


Figure 3.14. A contour map of the diode zero bias capacitance in units of 10^{-14} farads. Tick marks correspond to diode positions on the wafer.

grid almost functionless, because they are connected in parallel. To overcome this, a liquid crystal detection technique is developed to identify the shorted diodes [19]. Figure 3.15 is a photograph of a shorted diode that was found using this method. A layer of liquid crystal for 28-30 °C range is spun onto the wafer. Then current is injected into the rows that are shorted. Because the shorted diodes draw most of the current and therefore dissipate most of the heat, the color of the liquid crystal changes from red to blue as the temperature rises within the vicinity of the shorted diode. Once the short is found, an ultrasonic probe is used to remove the defective diode.

3.3 Test Fixture

Figure 3.16 shows a section of the diode-grid fabricated on a GaAs wafer. Figure 3.17 shows the diode-grid, glued on a glass slide with photoresist at the edge of a pc-board. This is convenient because the photoresist can be dissolved in acetone quite easily. There are a total of 50 DC bias lines available. Two edge connectors are used to feed the bias lines from the variable bias controller, which consists of an array of variable resistors driven by programmmable constant current sources. Electronic relays are used to provide low-frequency modulation of the DC bias for situations that require a reference signal.

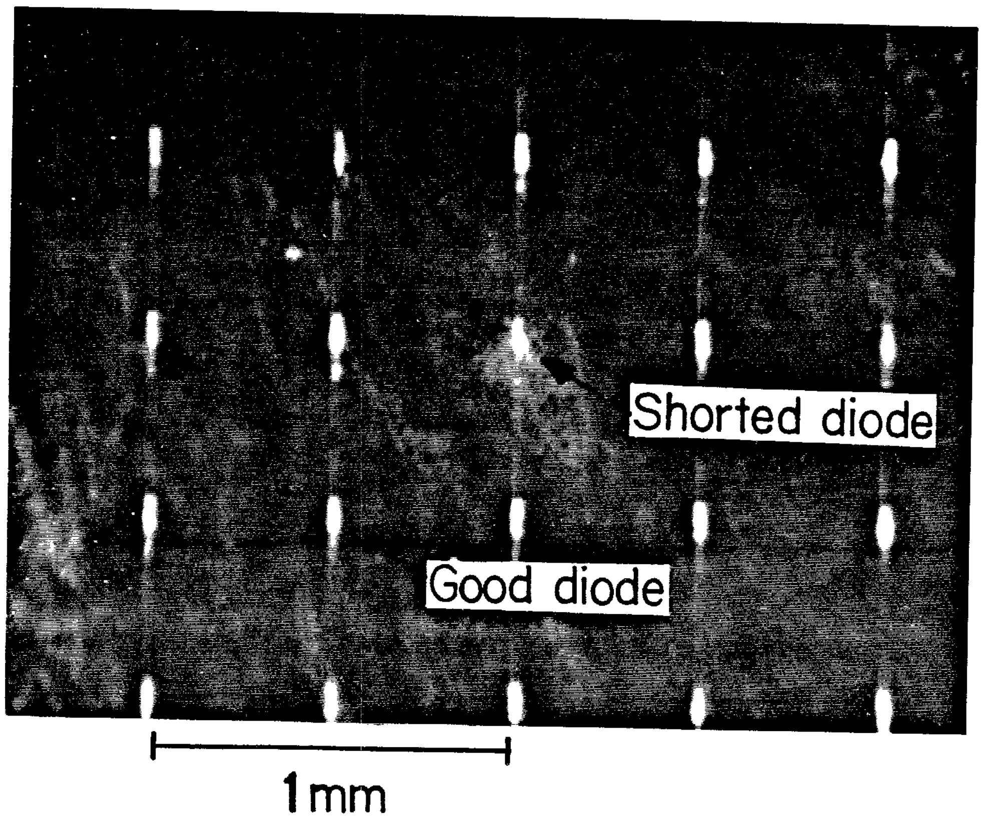


Figure 3.15. Photograph of a shorted diode found by a liquid crystal detection technique.

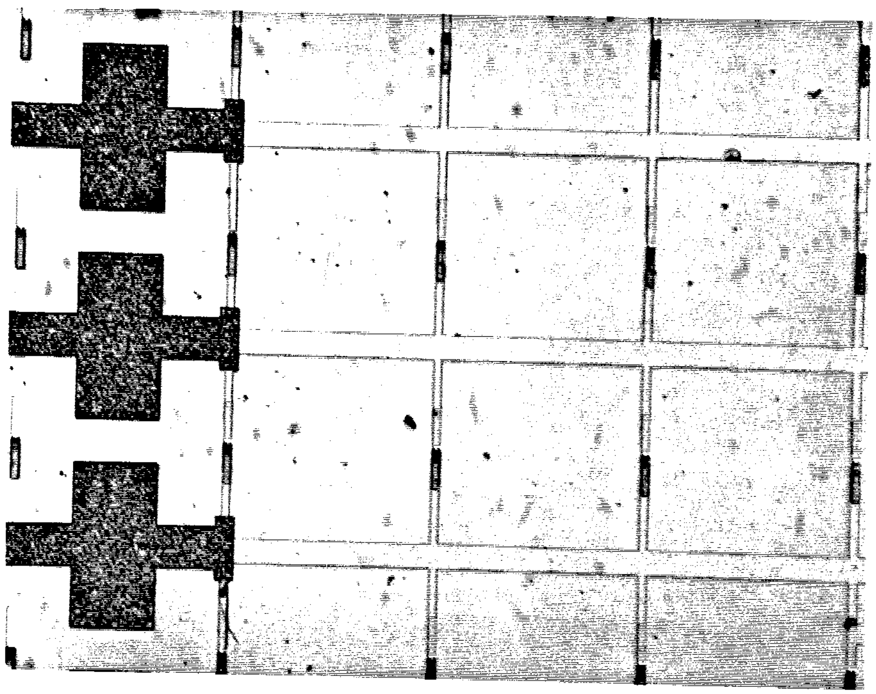


Figure 3.16. Photograph of a diode-grid with AuGe/Ni/Au bonding pads.

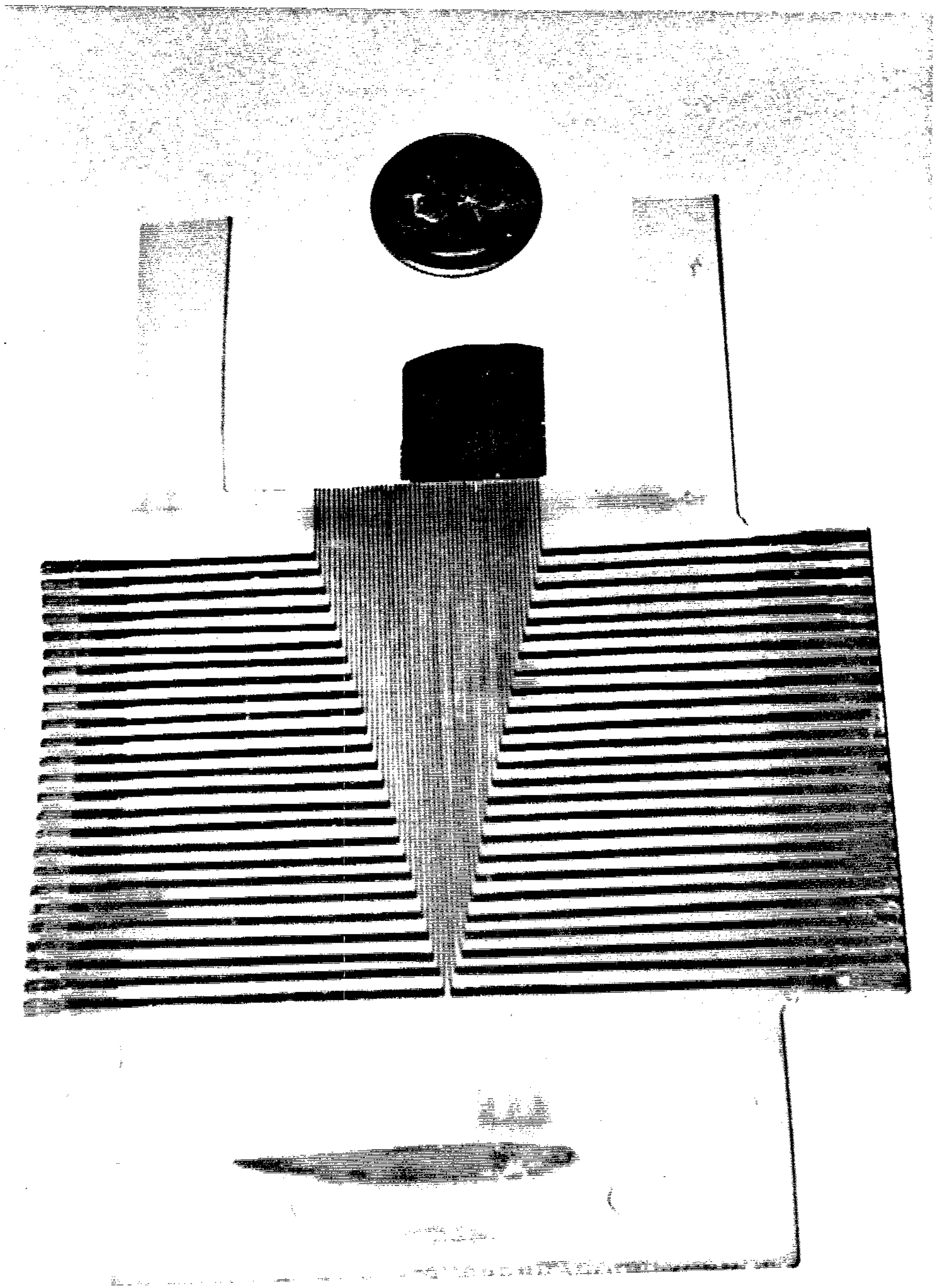


Figure 3.17. Photograph of a diode-grid mounted on a pc-board with 50 DC bias lines available.

References

[1] C. E. Zah, "Millimeter-Wave Monolithic Schottky Diode Imaging Arrays," *Ph. D. Thesis*, Chap. 3, pp.38–51, California Institute of Technology, California, 1986.

[2] M. H. Norwood and E. Shatz, "Voltage Variable Capacitor Tuning: A Review," *Proceedings of the IEEE*, **Vol. 56**, pp. 788–798, 1968.

[3] K. Lundien, R. J. Mattauch, J. Archer, and R. Malik, "Hyperabrupt Junction Varactor Diodes for Millimeter-Wavelength Harmonic Generators," *IEEE Trans. on Microwave Theory and Tech.*, **MTT-31**, pp. 235–238, 1983.

[4] R. S.Muller and T. I. Kamins, *Device Electronics for Integrated Circuits*, **Chap. 2**, pp. 65–104, John Wiley & Sons, Inc., New York, 1977.

[5] C. R. Hilsum, "Simple empirical relationship between mobility and carrier concentration," *Electronics Letters*, **Vol. 10**, pp. 259–260, 1974.

[6] S. M. Sze and G. Gibbons, "Avalanche Breakdown Voltages of Abrupt and Linearly Graded p-n Juntions in Ge, Si, GaAs, and GaP," *Appl. Phys. Lett.*, **Vol. 8**, pp. 111–113, 1966.

[7] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, **Chap. 4**, pp. 102–130, Cambridge University Press, New York, 1986.

[8] P. Penfield, Jr., and R. P. Rafuse, *Varactor Applications*, **Chap. 4**, pp. 57–91, Massachusetts Institute of Technology Press, Massachusetts, 1962.

[9] S. J. Eglash, M. D. Williams, P. H. Mahowald, N. Newman, I. Lindau, and W. E. Spicer, "Aluminum Schottky Barrier Formation on Arsenic Capped and Heat Cleaned MBE GaAs(100)," *J. of Vac. Sci. Technol.*, **Vol. B2(3)**, pp. 481–486, 1984.

[10] T. C. Leonard, "Prediction of Power and Efficiency of Frequency Doublers

Using Varactors Exhibiting a General Nonlinearity," *Proceedings of the IEEE*, **Vol. 51**, pp. 1135–1139, 1963.

[11] M. Binet, "Fast and Non-destructive Method of C(V) Profiling of Thin Semiconductor Layers on an Insulating Substrate," *Electronics Letters*, **Vol. 11**, pp. 580–581, 1975.

[12] A. Y. Cho and P. D. Dernier, "Single-crystal-aluminum Schottky-barrier diodes prepared by molecular-beam-epitaxy (MBE) on GaAs," *J. of Appl. Phys.*, **Vol. 49**, pp. 3328-3332, 1978.

[13] M. Heiblum, M. I. Nathan, and C. A. Chang, "Characteristics of AuGeNi Ohmic Contacts to GaAs," *Solid State Elect.*, **Vol. 25**, pp. 185–195, 1982.

[14] R. E. Williams, *Gallium Arsenide Processing Techniques*, **Chap. 11**, pp. 225–257, Artech House, Inc., Massachusetts, 1984.

[15] N. Yokoyama, S. Ohkawa, and H. Ishikawa, "Effects of the Heating Rate in Alloying of Au-Ge to n-Type GaAs on the Ohmic Properties," *Japan J. of Appl. Phys.*, **Vol. 14**, pp. 1071–1072, 1975.

[16] J. P. Donnelly and F. J. Leonberger, "Multiple-Energy Proton Bombardment in n^+-GaAs," *Solid-State Electronics*, **Vol. 20**, pp. 183–189, 1976.

[17] Y. Sato, M. Uchida, K. Shimada, M. Ida, and T. Imai, "GaAs Schottky Barrier Diode, ECL-1314," *Rev. of the Electrical Communication Lab.*, **Vol. 18**, pp. 638–644, 1970.

[18] M. Ida, M. Uchida, K. Shimada, K. Asai, and S. Ishida, "Fabrication Technology of Stable Schottky Barrier Gates for Gallium Arsenide MESFETS," *Solid-State Electronics* **Vol. 24**, pp. 1099–1105, 1981.

[19] J. L. Fergason, "Liquid Crystal in Nondestructive Testing," *Applied Optics*, **Vol. 7**, pp. 1729–1737, 1968.

Chapter 4

Diode-Grid Phase Shifter Measurements

In proposing a new design, the experimental procedure for testing the assumed model is also necessary. In the process of developing the experimental procedure, what is available in the laboratory often plays a role in deciding the experimental method. This chapter begins with a survey of possible existing methods of measurement for testing the diode-grids, and concludes that they are not suitable in our work. This is mainly due to the fact that wafers available to do our experiments were usually small and irregular in shape. Consequently, a small aperture reflectometer that uses a wave-front division interference technique was developed to measure the reflection coefficient of the diode-grid. The validity of this method is illustrated by reflection measurements of thin-film bismuth on fused quartz and its limitations are indicated by reflection measurements of fused quartz. The experimental procedure consists of curve-fitting the measured RF reflection coefficient with an equivalent circuit model based on transmission-line theory and comparing the best-fitted parameters with the corresponding parameters measured at low frequency.

4.1 Survey of Possible Experimental Methods

One of the possible methods of testing the diode-grid is based on simulation of a planar periodic array in waveguide [1]. Such simulation permits the use of a few elements in a waveguide to represent a large number of elements in an infinite periodic array. In the waveguide simulator, the array impedance can be measured. However, precision waveguide machining and sample mounting are required to duplicate the details of the array. Furthermore, a scaled model is probably required as the frequency approaches 100 GHz, and it is more desirable if the measurement can be made directly on the diode-grid itself. Therefore, methods based on quasi-optical techniques become more attractive.

In principle, various quasi-optical methods for measuring the complex dielectric constant of a material can be extended to test the diode-grid. They have demonstrated remarkable accuracy in dielectric measurements at millimeter wavelengths. The semi-confocal open-resonator [2] and quasi-optical network analyzer [3,4] produce results with uncertainty of about 1 % in the real part of dielectric permittivity and 10 % in the loss tangent. The Mach-Zehnder interferometer [5], and Michelson interferometer [6] provide five or six significant figures for refractive index and 1 % accuracy in loss tangent.

In the semi-confocal resonator method, a sample free of any attachments is mounted on the planar mirror side of the resonator, and a change in the Q of the resonator and the resonant frequency are used to determine the complex dielectric permittivity. Jones [2] successfully demonstrated precise dielectric measurements at 35 GHz. The sample should be plane-parallel with a diameter $\geq$ 5 cm. His sample ranged from 7 cm to 7.5 cm. The minimum spot size of the beam determines how small a sample can be used, and at 100 GHz samples with a diameter larger than 3 cm are required. This was too large for us.

In the other three approaches, various optical configurations for wavefront interference are used to assess the dielectric permittivity. Their general principle of operation is interesting and will simply be illustrated with the Michelson interferometer as shown in figure 4.1. An incident wave is divided into two parts by the beam splitter. One part of this wave goes to a scanning mirror and becomes the reference wave. The other part goes to the sample, reflects off the sample, and becomes the signal wave. Then the two waves recombine at the beam splitter and go to the detector. The measured intensity of the interfering signals allows one to find the reflection coefficient of the sample. To calibrate the system, the

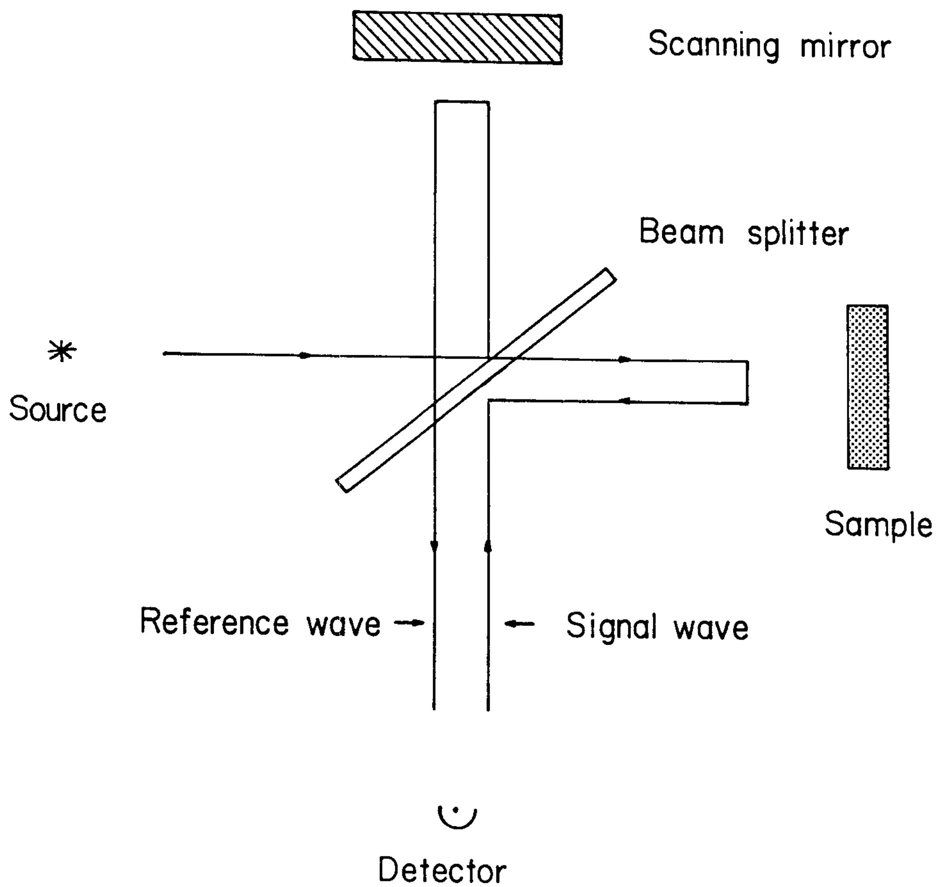


Figure 4.1. Schematic representation of a Michelson interferometer.

microwave technique of using an open, a short, and a matched-load can be used. Placing a planar mirror at certain position along the horizontal arm of the interferometer allows one to define a reflection coefficient of -1. Translating the mirror by a quarter-wavelength allows one to define a reflection coefficient of $+1$. And removing the mirror allows one to define a zero reflection coefficient.

Figure 4.2 shows an actual setup of the Michelson interferometer. Teflon lenses were included to collimate and focus the beam onto the diode-grid since the grids were typically small (2 cm by 2 cm). Using results published by Harvey [7], we designed an artificial quarter-wave matching layer in the form of groves to match the surfaces of the dielectric lens. Also, absorbing materials were strategically placed to minimize any stray reflections. Despite these efforts, the measured results were inconsistent with transmission-line theory because the calibration procedure was not sufficient to calibrate out those extraneous reflections coming from the dielectric lens and absorbers in the neighborhood of the diode-grid. Although more complete calibration procedures are available [6], they are very sophisticated for practical use. In addition, these methods require samples with large lateral dimensions. For example, in the multiport reflectometer [3], Stumper used samples 7 cm to 8 cm in diameter to make reflection measurements at 392 GHz. These are the factors that make the small aperture reflectometer attractive for samples that are small and irregular in shape.

4.2 Small Aperture Reflectometer

Figure 4.3 shows a small aperture reflectometer developed to measure the reflection coefficient of the diode-grid. The idea is to use an absorbing screen with a hole in the center to divide an incident wave-front into two parts. The wave that reflects off the absorber is the reference, while the other part reflecting off the sample is the signal. The interference of these two reflected waves is

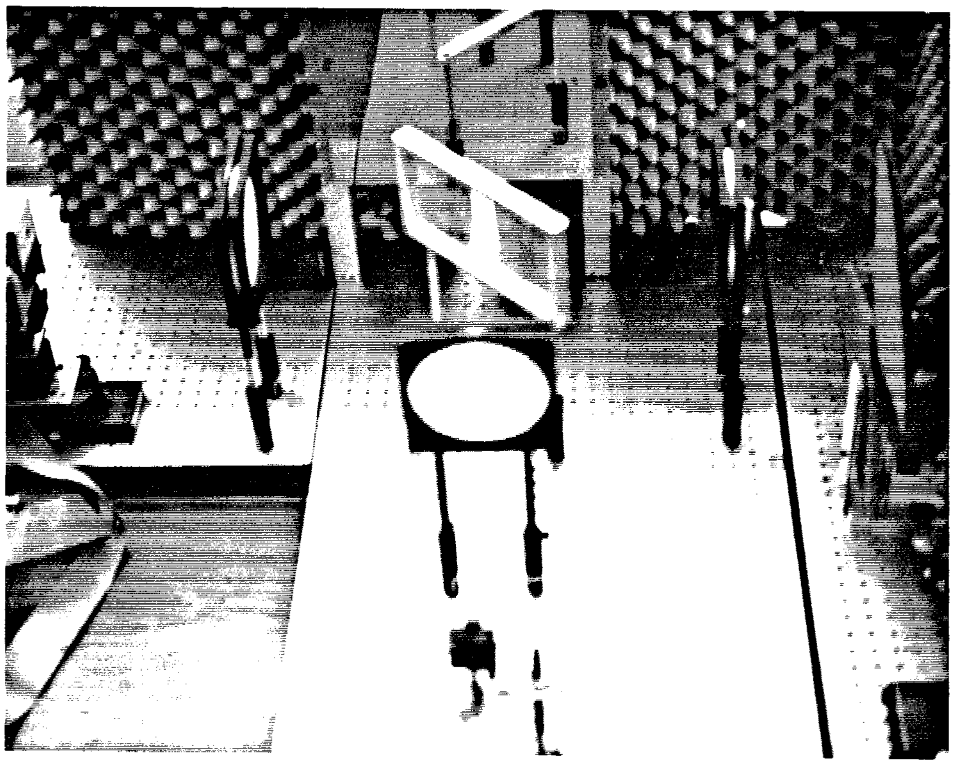


Figure 4.2. Photograph of a Michelson Interferometer for millimeter waves.

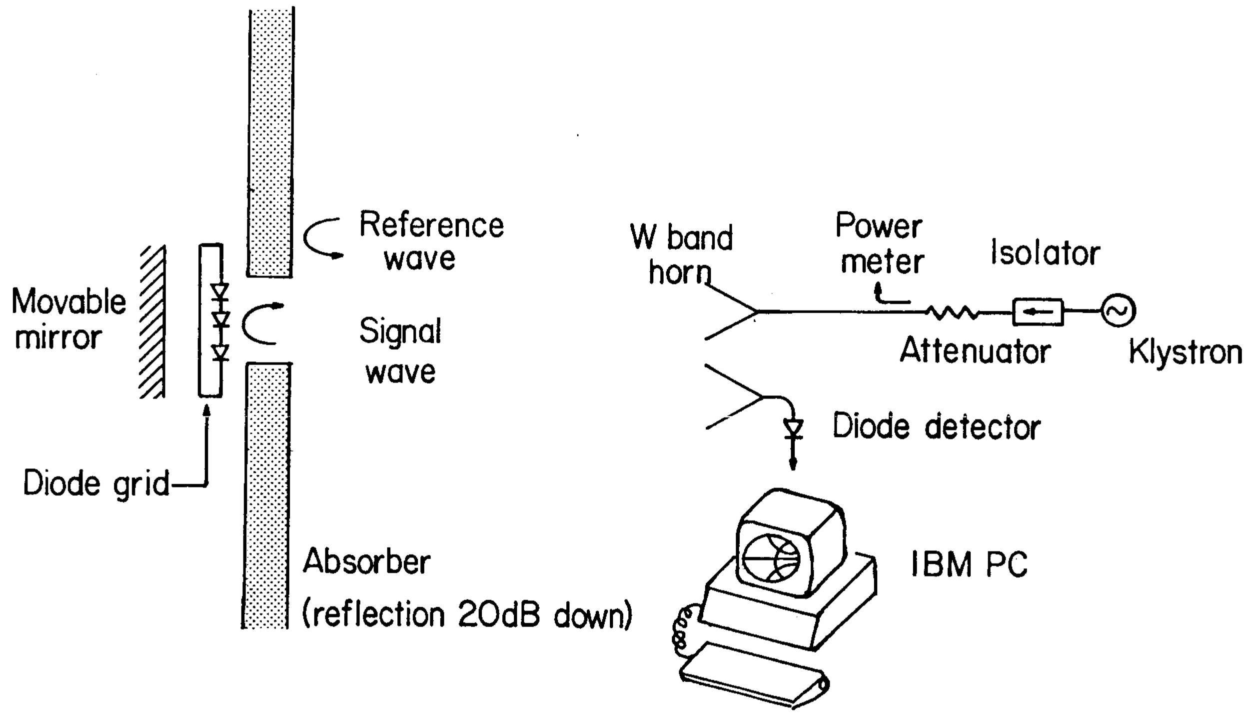
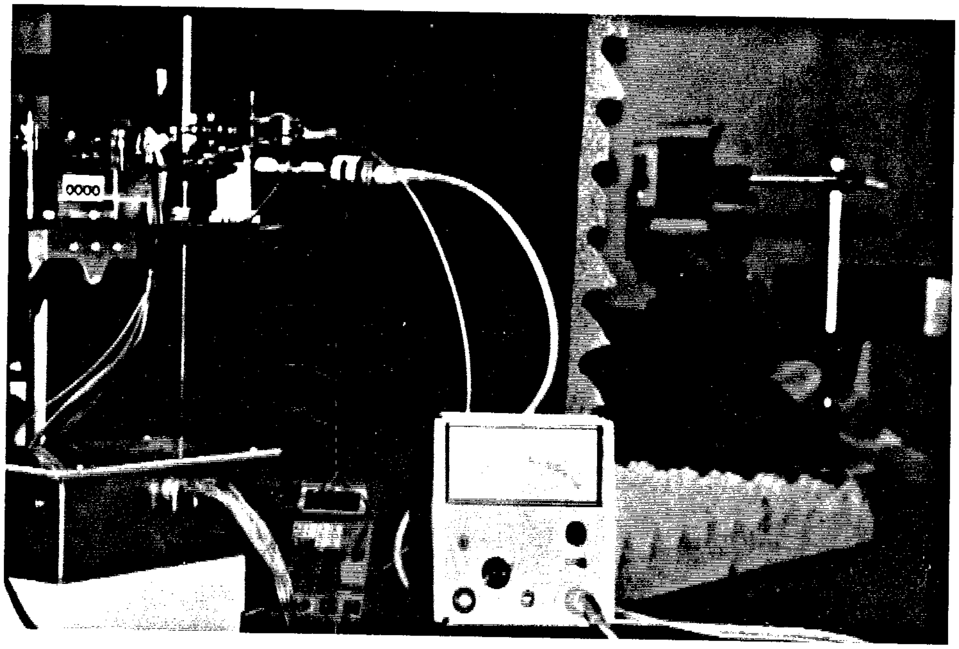


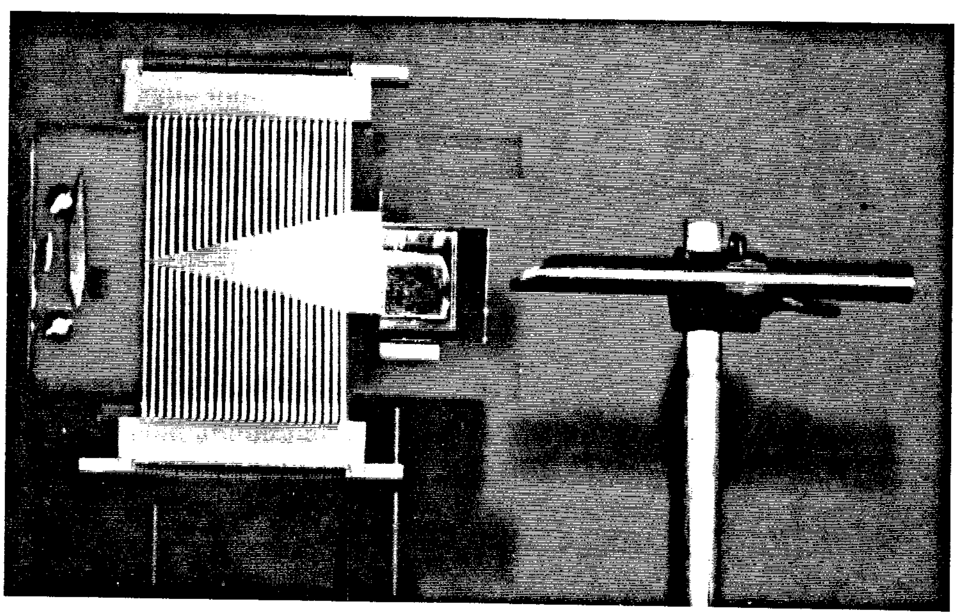
Figure 4.3. Schematic representation of a small aperture reflectometer.

measured as the sample is translated relative to the absorber. An absorbing screen is chosen to approximately balance the energy in the two reflected waves so that the measured intensity has sufficient contrast. The phase and amplitude of the reflection coefficient of the sample can be found by least-square fitting each interference pattern or by a simple four-point method [8]. The scanning mirror serves as a tuning parameter as well as a standard load for calibration. The system alignment and calibration can be done quite quickly, and the measurement can be computerized.

Figure 4.4 shows an actual setup for the small aperture reflectometer. Initially a Varian klystron source with 100 mW of output power was used, but later it was found that a Hughes Gunn-diode source with 10 mW of output power was sufficient to do the measurement. The input power is sampled by a 10-dB directional coupler and monitored by an HP432A power meter thru a Hughes thermistor head. The frequency is measured by a Hughes wave-meter before and is checked after the experiment to determine the appropriate step size and to ensure that no significant frequency shift occurs. The measured input powers are used to normalize the measured intensities. The area of the transmitting horn is 4 cm^2. The intensity of the interference pattern is measured by a receiving horn that feeds into a zero bias detector. The output is a DC signal, which is monitored by a HP 3478A multimeter. An IBM-PC is used to control the equipment. It uses a Capital Equipment interface board to communicate with HP equipment and a Data Translation A/D and D/A converter to control the stepper motors, which have a resolution of 1 μm. Figure 4.5a shows the translation stage. Figure 4.5b shows the absorbing screen that is placed next to the diode-grid, which is located at a distance of 82 cm away from the horns. This corresponds to six times the far field condition. To accommodate different sizes and shapes of diode-grid, a small replaceable template of absorber is used in conjunction with a bigger

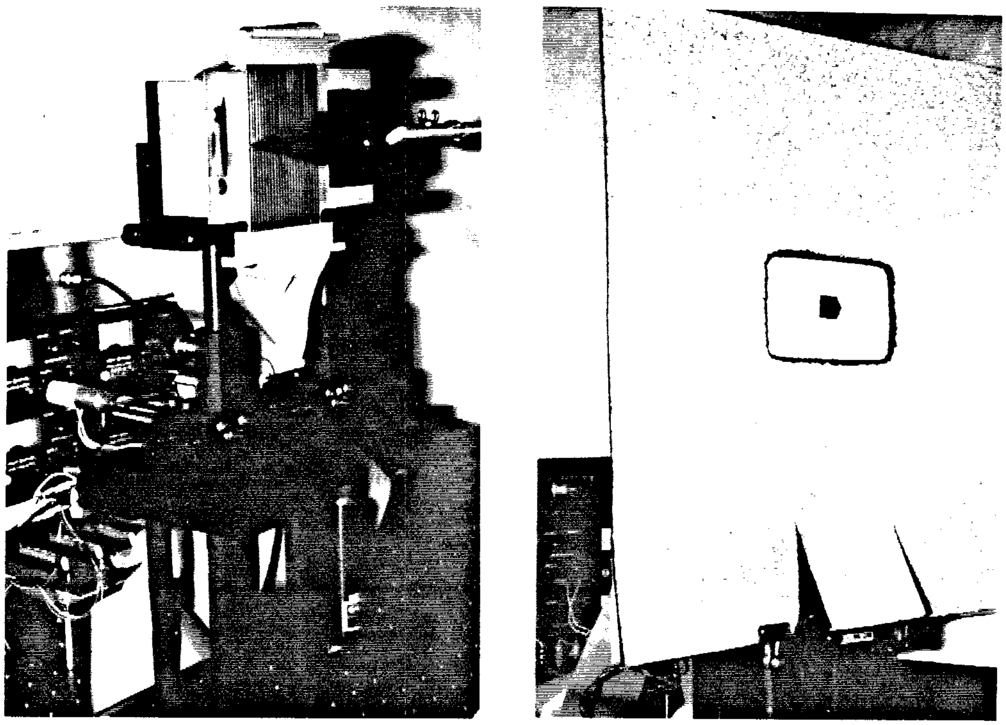


(a)



(b)

Figure 4.4. Photographs of the small aperture reflectometer setup. (a) Overall view. (b) Close-up view of the test fixture and minature microscope.



(a) (b)

Figure 4.5. (a) Sideview of the computer-controlled translation stage. (b) The absorbing screen that is used to divide an incident wavefront in the small aperture reflectometer.

absorbing screen. Both the diode-grid and the scanning mirror are aligned with a Helium-Neon laser. The mirror is made by evaporating 2 μm of gold on glass. The initial distance between the sample and the scanning mirror is measured by a miniature microscope with a resolution of 25 μm.

The effect of interference can be demonstrated by plotting the power detected by the diode detector as a function of the mirror position. Figure 4.6 shows that the received power varies sinusoidally with the mirror position. The maxima correspond to constructive interference, while the minima correspond to destructive interference. Physically, more energy is deflected at the peaks and less energy is deflected at the valleys into the field of view of the detector. Since the reflection coefficient of the sample is proportional to the complex amplitude of the interference pattern, one way to find the amplitude and phase is simply to fit a sinusoidal curve through the measured data. This is shown in figure 4.6 with a solid line. However, the curve-fitting process is time-consuming.

Figure 4.7 shows pictorially another method in which both the amplitude and the phase of a sinusoid can be calculated. This is called the four point method, and it is based on simple phasor trigonometry. It was developed by Wyant [8] in optical interferometry for three-dimensional sensing. When the intensity of the sinusoid is sampled four times consecutively at 90° intervals, both the amplitude and phase of the sinusoid can be calculated by the law of tangents and the Pythagorean theorem. The 90° phase shifts can be introduced in the form of optical path delay by translating the sample relative to the screen at intervals of one-eighth of a wavelength. Both the curve-fitting method and the four-point method have been used, and they agree to within 2 % for amplitude and 3° for phase. The four-point method is preferred since it is faster. Also, it is interesting to note that the use of the four-point method in the small aperture reflectometer makes the system analogous to a six-port network analyzer. However, in the small

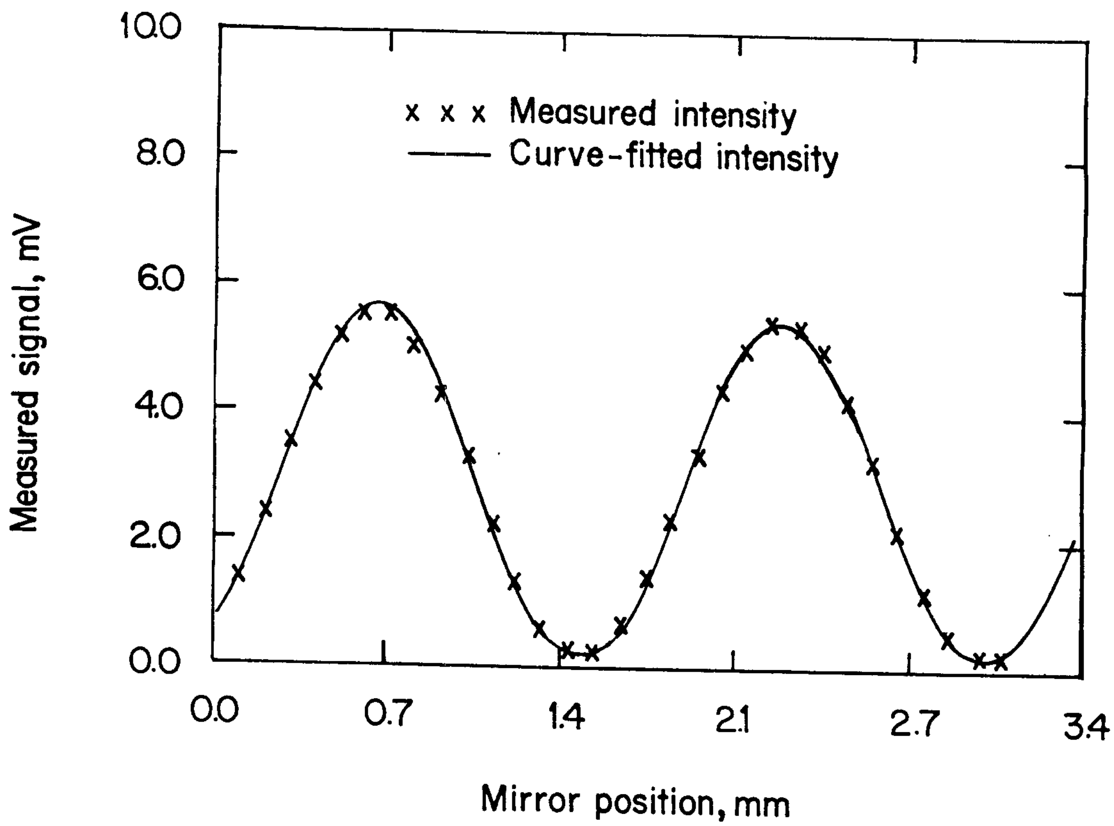


Figure 4.6. An interference pattern measured at 93 GHz.

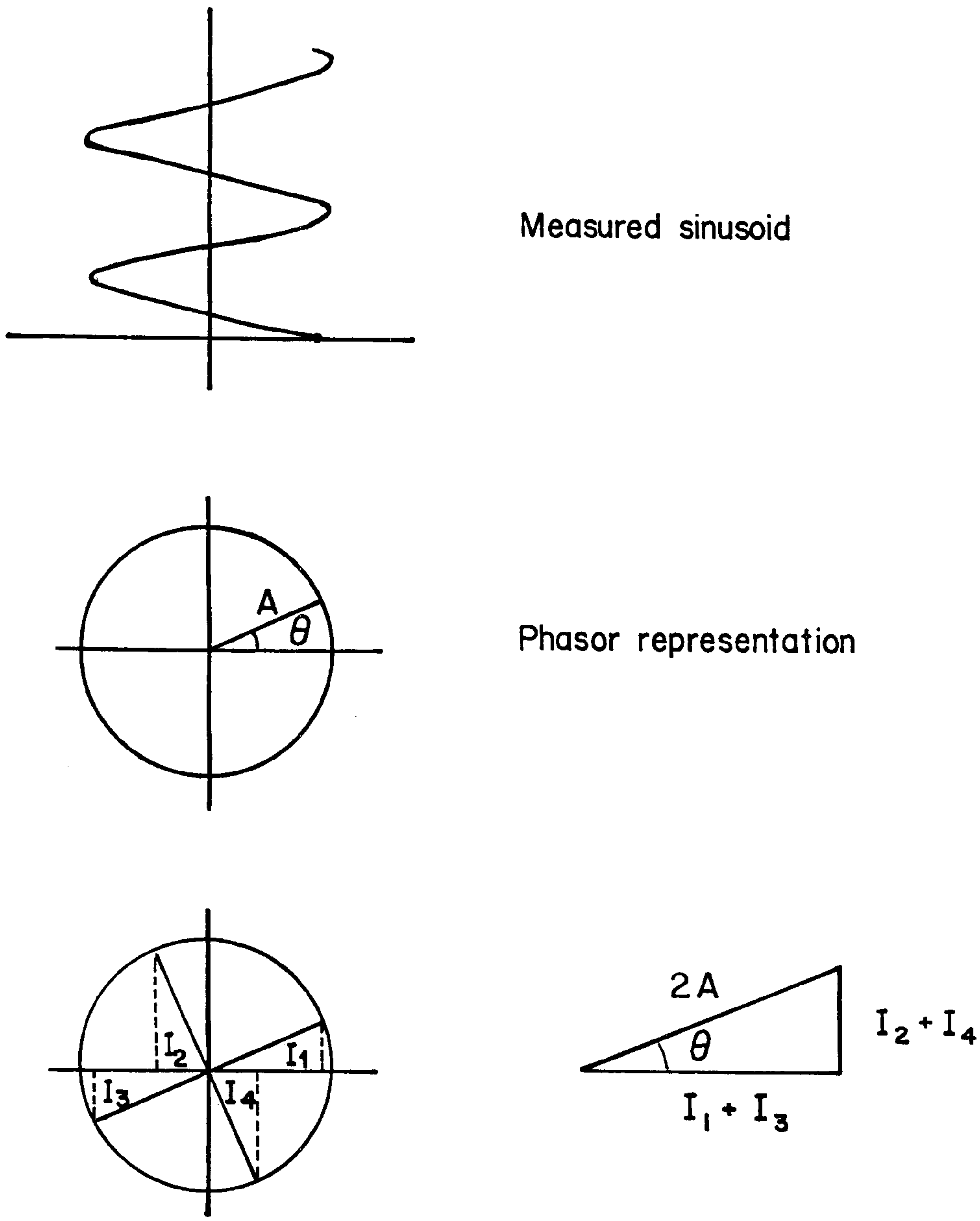
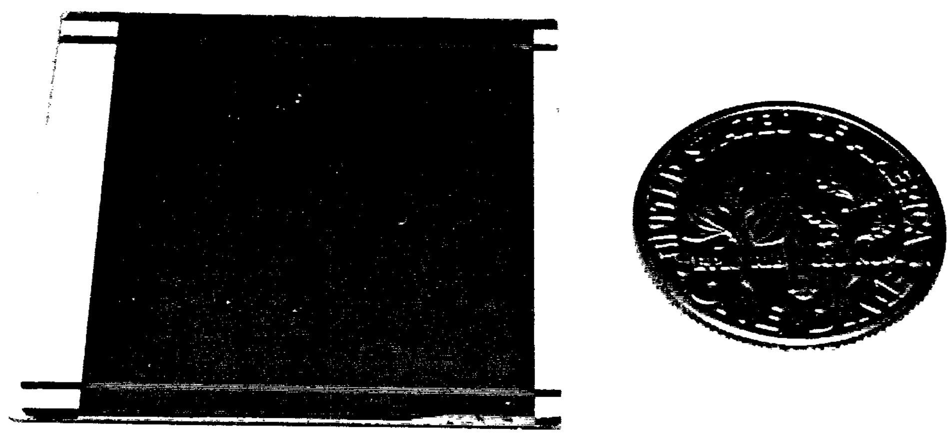


Figure 4.7. A pictorial illustration of the four-point method.

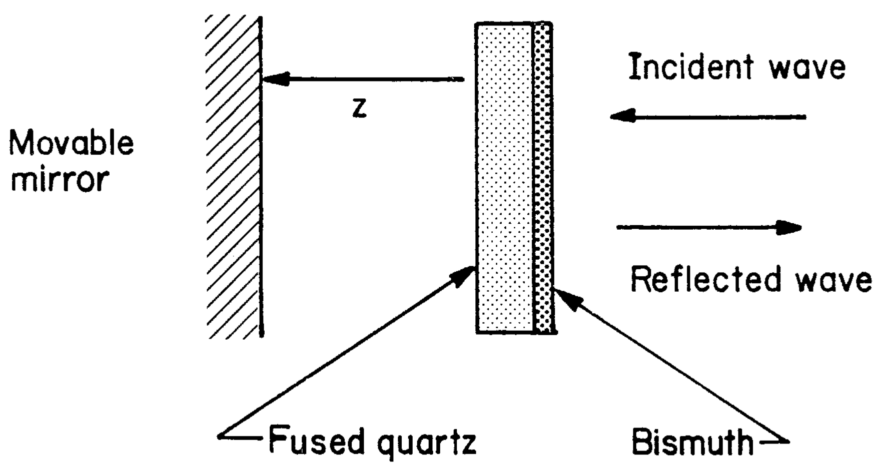
aperture reflectometer only one detector is used, while in the six-port network analyzer four detectors are required.

Reflection Measurements of Bismuth on Fused Quartz

The validity of the small aperture reflectometer can be illustrated with the reflection coefficient measurement of thin-film bismuth since we can control precisely the way in which the sample is prepared. Figure 4.8a shows a one-square inch thin film bismuth deposited on a 3 cm square fused-quartz plate by electron-beam evaporation in a diffusion pumped vacuum system. The bismuth thickness is 608 Å. Using the four-point probe procedure, we measured the sheet resistance the bismuth film via four gold deposited strips at the edge of fused quartz to be 92.2 Ω. The fused quartz material is Dynasil # 4000. It is plane-parallel to within 5 μm. Figure 4.8b shows the configuration in which the reflectance and phase of reflection were measured at 93 GHz. Figure 4.9 shows the result of the reflection measurement or the tuning curve. The reflectance reaches a maximum of 100 % at a mirror position of 720 μm. This is primarily to due the effect of the mirror, which basically presents an electrical short at the plane of the bismuth film. At a mirror position of 1590 μm, the effect of the mirror is equivalent to an open circuit at the plane of the bismuth film; therefore, the reflectance reaches a minimum. The value of this minimum is determined by the sheet resistance. Theoretical curves are plotted using the measured bismuth sheet resistance (92.2 Ω), fused quartz thickness (434 μm), index of refraction (1.96 from Afsar and Button [6], initial mirror position (203 μm), and a best-fitted length parameter due to phase calibration (2067 μm). The phase-calibration length is the distance between the input surface of the sample (bismuth film) and a reference plane at which the measured phase of reflection is 180° for the mirror. The phase-calibration length was measured to be 2019 μm, which disagrees with the best-fitted value by 48 μm



(a)



(b)

Figure 4.8. (a) Photograph of thin-film bismuth on fused quartz. (b) Reflection measurement configuration of bismuth at 93 GHz.

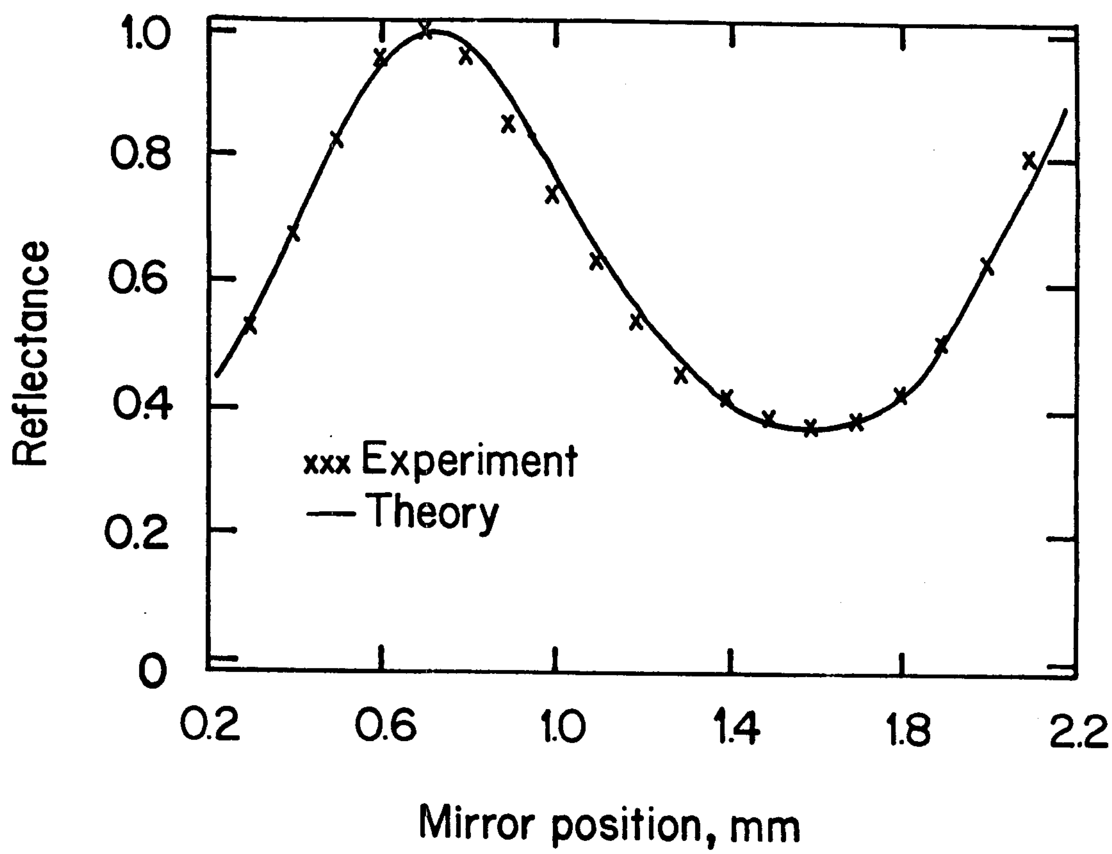


Figure 4.9a. Measured reflectance of thin-film bismuth on fused quartz at 93 GHz.

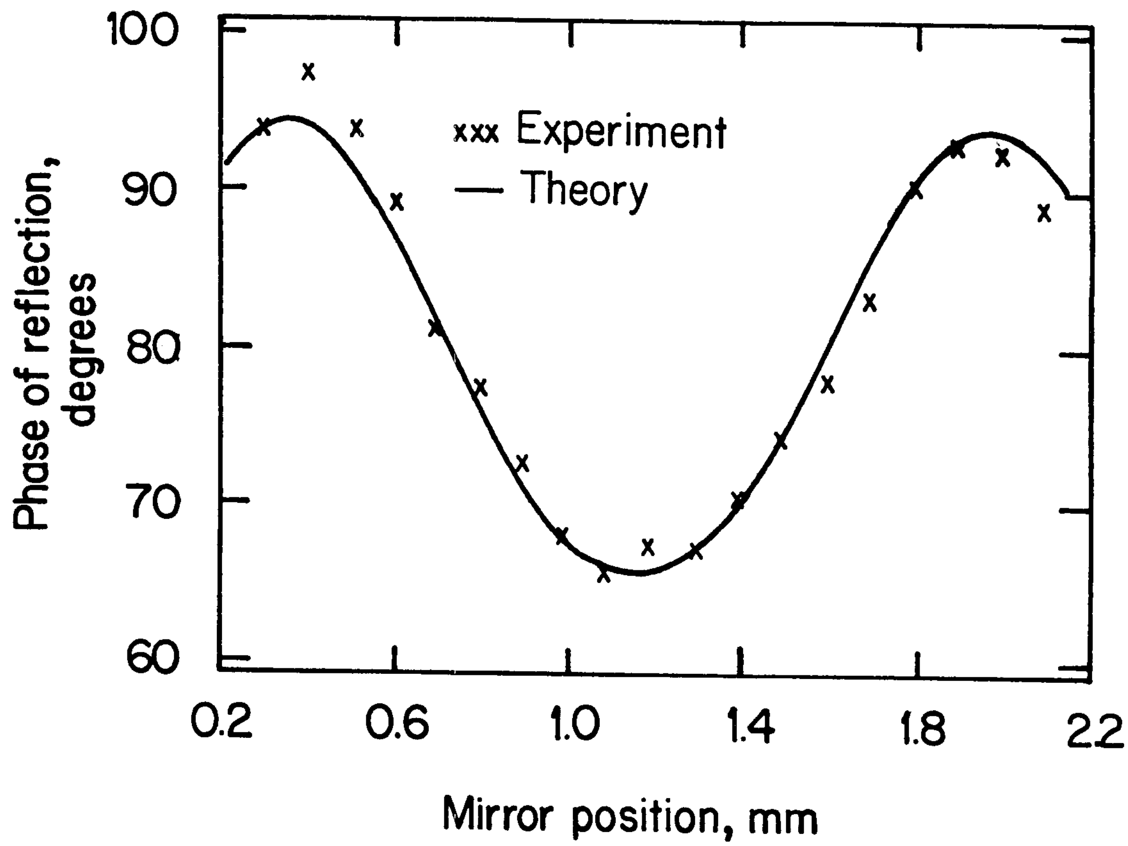


Figure 4.9b. Measured phase of reflection of thin-film bismuth on fused quartz at 93 GHz.

in physical length or 10° in electrical length. This is about as accurate as we can measure the phase-calibration length since, a miniature microscope with a resolution of 25 μm is used to measure distances from the side of the sample. When the sheet resistance of thin-film bismuth is also treated as a fitting parameter, the best-fitted value is 91.6 Ω. This agrees quite well with the measured sheet resistance (92.2 Ω) at DC.

4.4 Reflection Measurements of Fused Quartz

The limitation of the small aperture reflectometer is observed when it is used to measure low loss materials such as fused quartz. Figure 4.10 shows the configuration in which the reflection measurement was made at 89 GHz. The reflection coefficient is measured as the mirror is tuned. Figure 4.11a shows the measured reflectance. The calculated reflectance is plotted using the measured fused quartz thickness (434 μm), initial mirror position (203 μm), and refractive index for fused quartz (1.96 from Afsar and Button [6]). The calculation does not agree well with the measured reflectance when the absorption index for fused quartz (0.0005 from Afsar and Button [6]) is used. This is shown with a solid line. In order to get a better agreement, an absorption index of 0.02 must be used. This is shown in figure 4.11a with a dashed line. The dips in the measured reflectance are probably due to power leaking laterally. Figure 4.11b shows the measured phase, calibrated according to the measured phase calibration length (1105 μm). It is interesting to note that the calculated phase of reflection agrees quite well with the measured phase. An explanation for this is the following. If power escapes as described above and the amount of power loss is relatively constant during a reflection measurement scan, then the measured phase is relatively unaffected because the four-point method, which takes the ratio of a difference of four intensity measurements to calculate the phase, effectively calibrates the

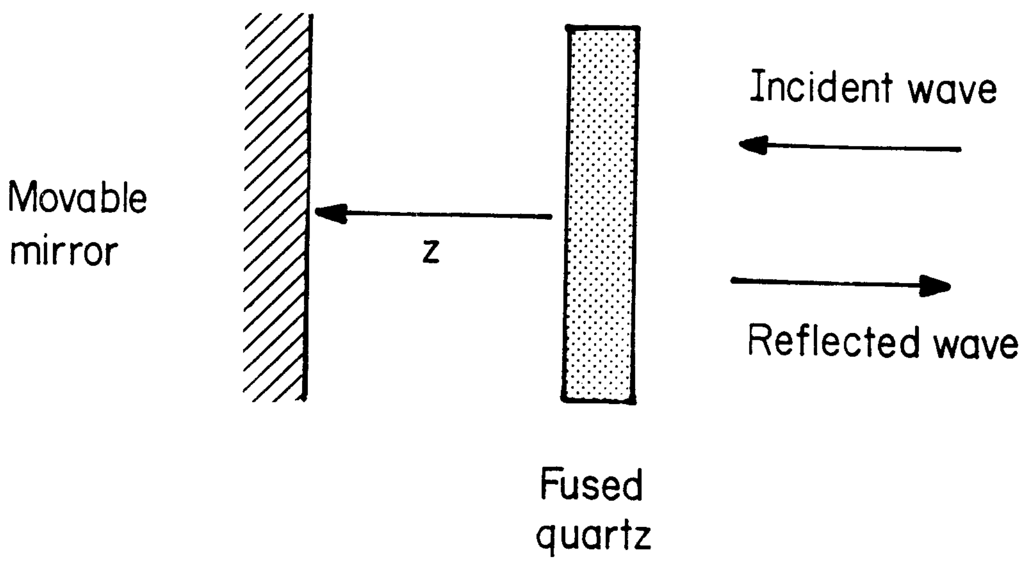


Figure 4.10. Reflection measurement configuration of fused quartz.

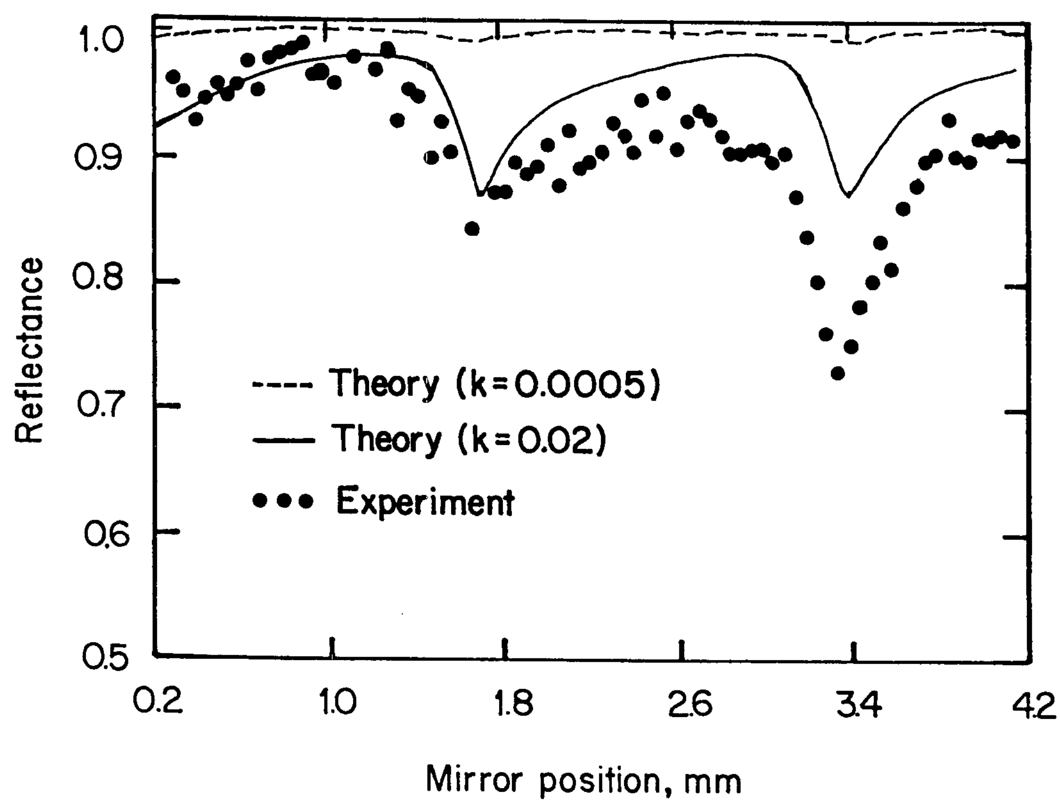


Figure 4.11a. Measured reflectance of fused quartz at 89 GHz.

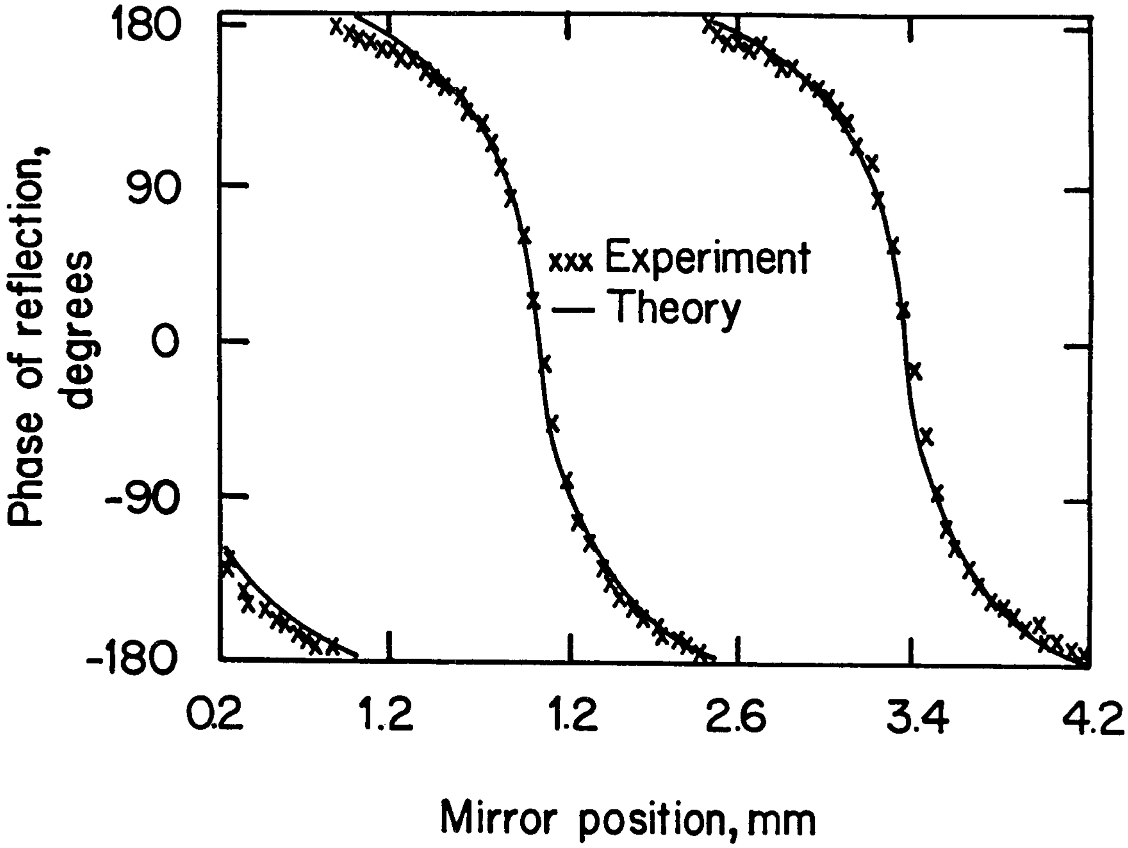


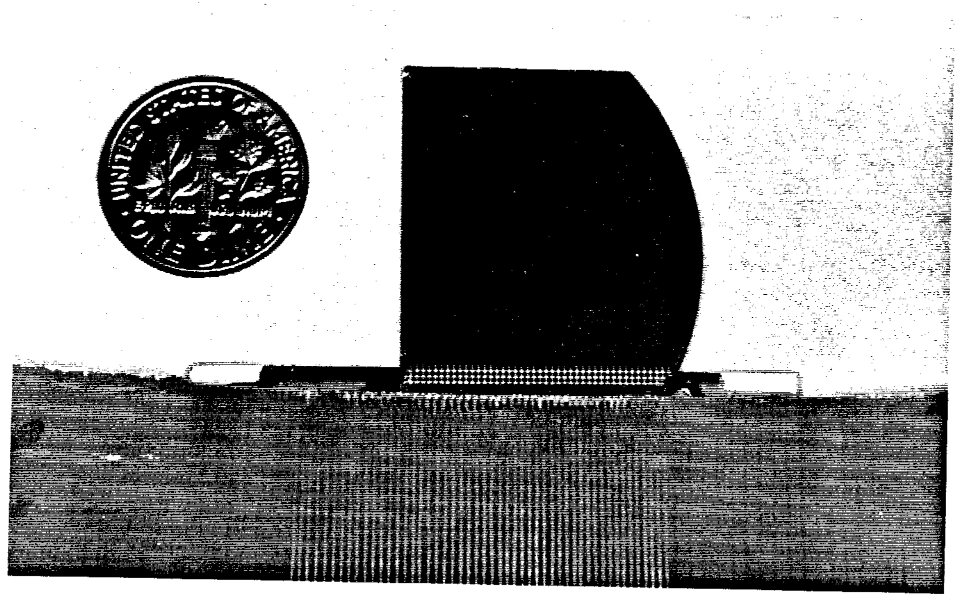
Figure 4.11b. Measured phase of reflection of fused quartz at 89 GHz.

loss of power out of the calculation. This also indicates that power is most likely leaking out from the etalon formed between the quartz and the mirror. Interestingly, no dips occurred in the previous reflectance measurement of thin film bismuth on fused quartz. An explanation for this is that the thin-film bismuth, being lossy, damps out most of the power during situations that favor lateral power leakage from the etalon.

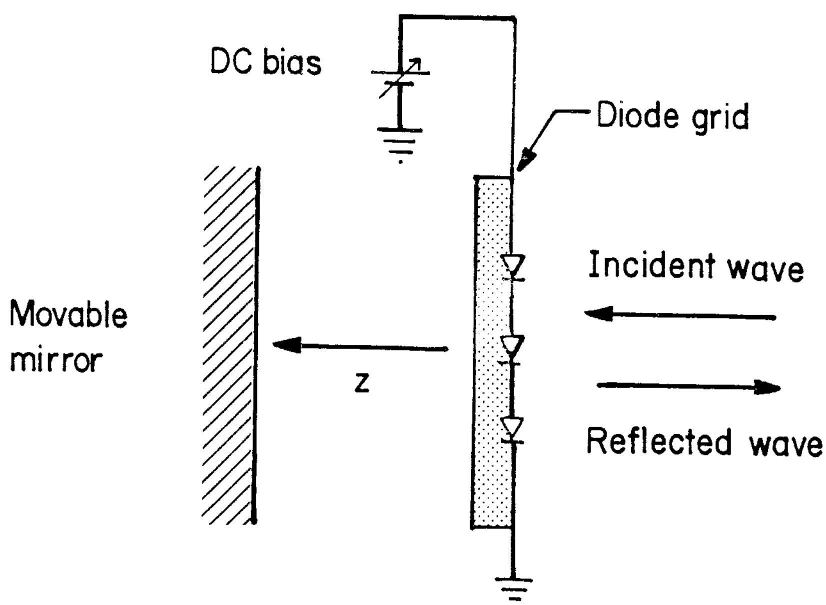
4.5 Reflection Measurements of Diode-Grids

Comparisons between theory and experiment up to this point have been reasonably good. This is because the quartz plate is plane-parallel to within $5\,\mu$m and has a uniform layer of bismuth. On the other hand, a diode-grid has both thickness variation and nonuniform device parameters. Typically, the wafer thickness varies between $10\,\mu$m and $30\,\mu$m. This is due to the manual lapping procedure. A total of 10 % of the diodes are expected to be open-circuited. This is due to over-etching during diode fabrication and removal of bad diodes that are shorted electrically or have a low breakdown voltage during diode testing. They tend to scatter randomly throughout the wafer. Nonuniformity of diode characteristics are also expected. They scatter less randomly throughout the wafer. This is mainly due to material properties and alignment during fabrication. These factors are not readily amenable to analysis and make a comparison between theory and experiment difficult. This is why sample preparation plays such a vital role in quasi-optical dielectric measurements [2-6]. Although our calculation does not take these variations into account, they should represent some sort of average and provide useful information for the designer.

A family of tuning curves were measured in order to see the effect of DC bias and millimeter frequency on the diode-grid circuit parameters. Figure 4.12a shows a diode-grid mounted on a pc-board that provides external DC bias to



(a)



(b)

Figure 4.12. (a) Photograph of a diode-grid mounted on a pc-board. (b) Reflection measurement configuration of a diode-grid.

the diodes. A network of variable resistors driven by constant current sources is used to provide adjustable floating voltages to each row of the diode grid. The physical dimensions of this wafer are 2 cm in width, 3 cm in length and 376 μm in thickness. Approximately 91 % of the varactors are functional and the rest are open-circuited. Figure 4.12b shows the configuration in which the tuning curves were made. Using a multi-dimensional simplex optimization algorithm [9], we curve-fitted the measured results with an equivalent circuit model based on a transmission-line theory. The error function was defined to be the absolute value of the difference between the measured and the calculated complex reflection coefficient.

Figure 4.13 shows one of these measured tuning curves at 94 GHz. This particular curve was measured with zero bias on the diode-grid. Four fitting parameters were used. The first parameter, phase calibration length, represents the thickness of a layer of air inserted in front of the diode-grid to account for the inaccuracy in phase-calibration. It allows the calculated phase of reflection to be adjusted by a constant offset. The second parameter, initial mirror position, represents the thickness of another layer of air inserted between the diode-grid and the mirror to account for the inaccuracy in measuring their initial separation. This has the effect of translating both the calculated reflectance and phase of reflection horizontally. The last two fitting parameters are the real and imaginary part of the diode-grid impedance. On this basis, theoretical curves were plotted using the best-fitted phase-calibration length (792 μm), initial mirror position (1663 μm), diode-grid impedance ($58 + j94\,\Omega$), the measured wafer thickness (376 μm), and the index of refraction of GaAs (3.6 from Afsar and Button [6]). The measured phase-calibration length was 727 μm and the initial mirror position was 1702 μm. The measured diode series resistance was 78 Ω with a standard deviation of 19 Ω and the zero bias capacitance was 30 fF with

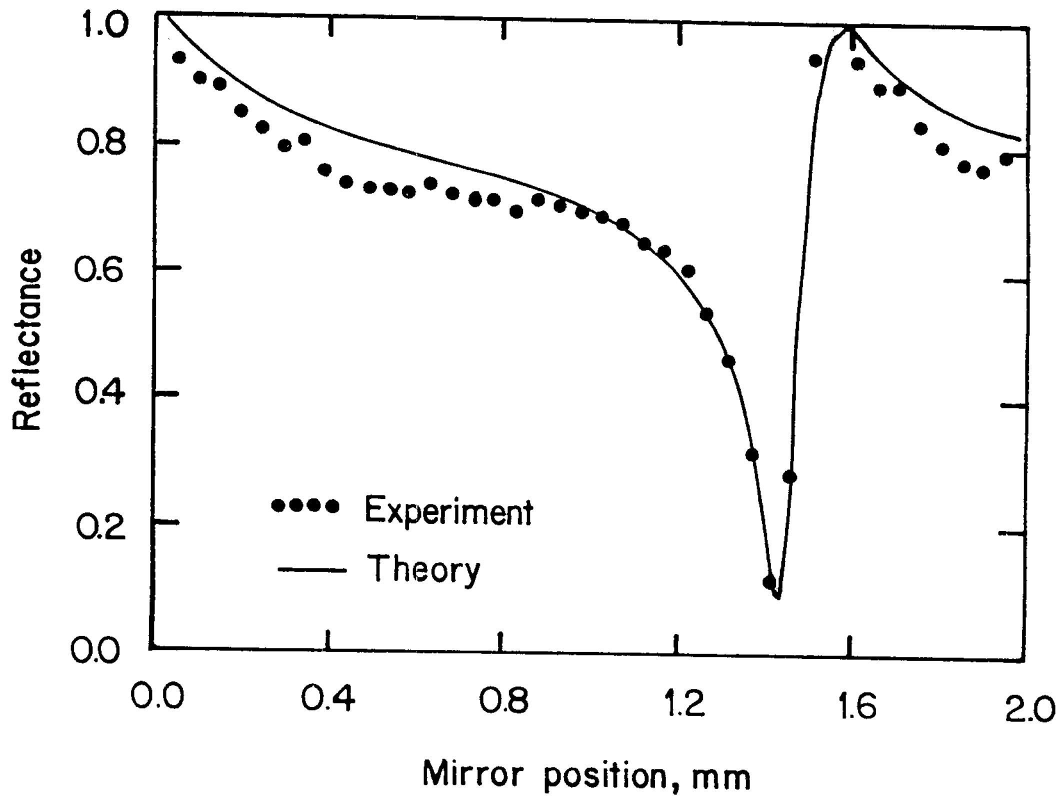


Figure 4.13a. Measured reflectance tuning curve of a diode-grid at 94 GHz.

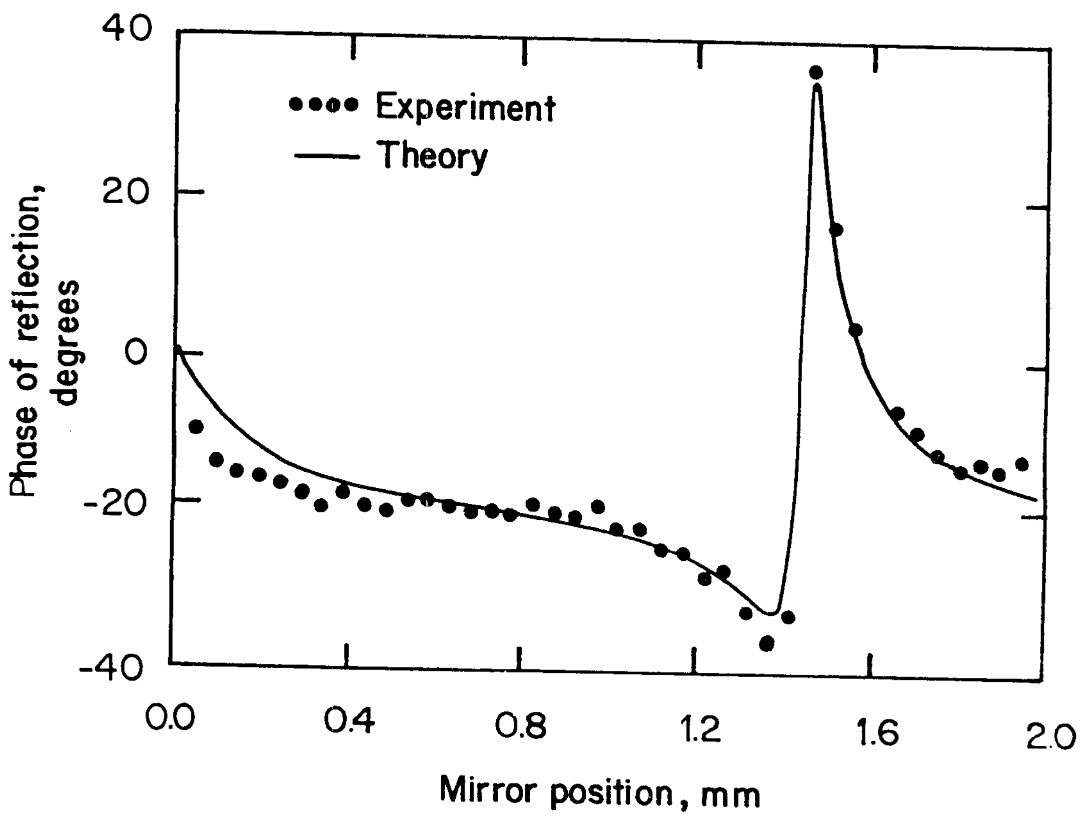


Figure 4.13b. Measured phase of reflection tuning curve of a diode-grid at 94 GHz.

a standard deviation of 10 fF. This is based on sampling 95 out of 2000 possible diodes. From the measured grid period (504 μm), strip width (18 μm), and strip length (450 μm), the strip inductance calculated from the quasi-static formula is 0.26 nH, which corresponds to an inductive reactance of 153 Ω at 94 GHz. Substracting the best-fitted diode-grid reactance from the calculated inductive reactance, we get 59 Ω for the capacitive reactance due to the varactor. This corresponds to 29 fF, which agrees with the measured zero-bias capacitance (30 ± 10 fF) at 1 MHz. Following this procedure of measuring diode capacitance at Rf frequency, figure 4.14a compares the capacitance-voltage (CV) measured characteristic at 94 GHz with the CV characteristic measured at 1 MHz. Figure 4.14b shows the corresponding series resistance measured at 94 GHz. The decreasing trend of the series resistance as a function of reverse bias is expected, since the resistance associated with the undepleted region of the diode decreases as the reverse bias increases; however, the amount of the decrement seems a little high.

To investigate the phase shift capability of this diode-grid, the measured phase and amplitude of these tuning curves for a particular mirror position is plotted as a function of bias voltage. The largest phase shift occurs at a mirror position of 3.23 mm. Figure 4.15 shows a comparison between experiment and theory based on transmission-line. In calculating the reflectance and phase of reflection, the assumed diode-grid parameters are based on average values of the corresponding parameters measured from the tuning curves. The average series resistance is 49 Ω, the minimum and maximum diode-grid reactance are 60 Ω and 105 Ω, respectively, and the average diode-grid to mirror separation is 3.16 mm. However, a phase calibration length of 795 μm is used. The calibration length averaged from the measured tuning curves is 761 μm, which corresponds to an 8° vertical shift. Phase shift performance is about 40° and average reflection loss is 6-dB.

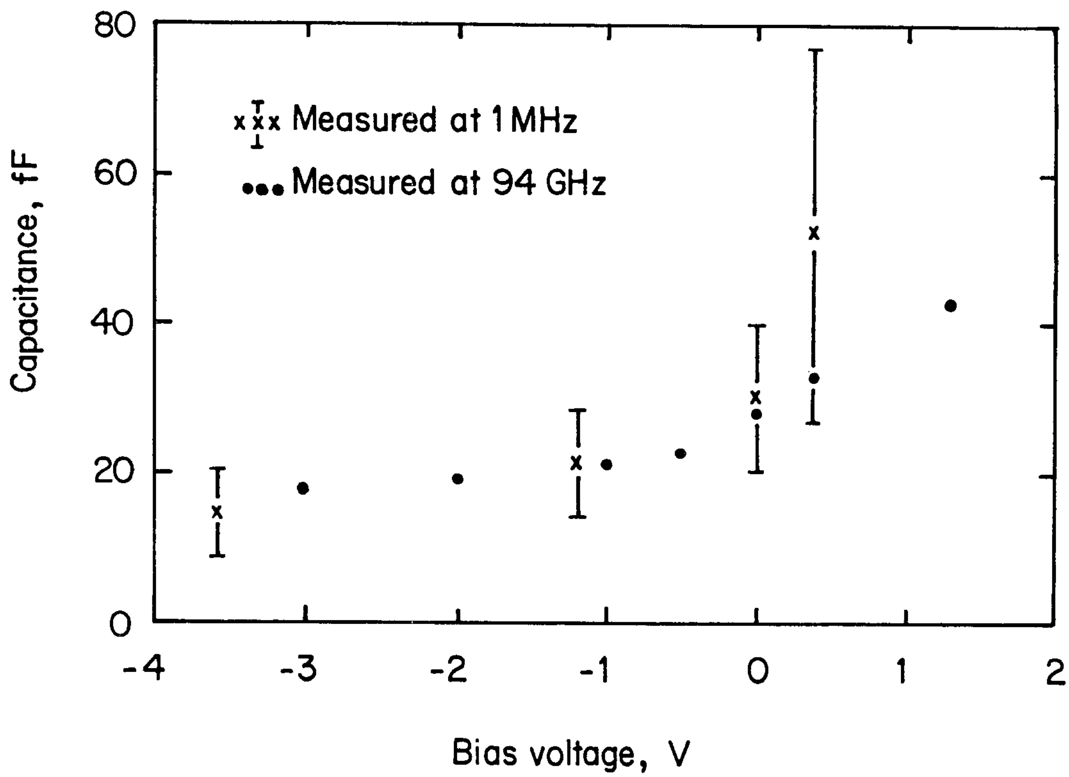


Figure 4.14a. A comparison between the measured capacitance-voltage (CV) characteristic of a diode-grid at 94 GHz and the measured CV characteristic at 1 MHz. Bars represent one standard deviation.

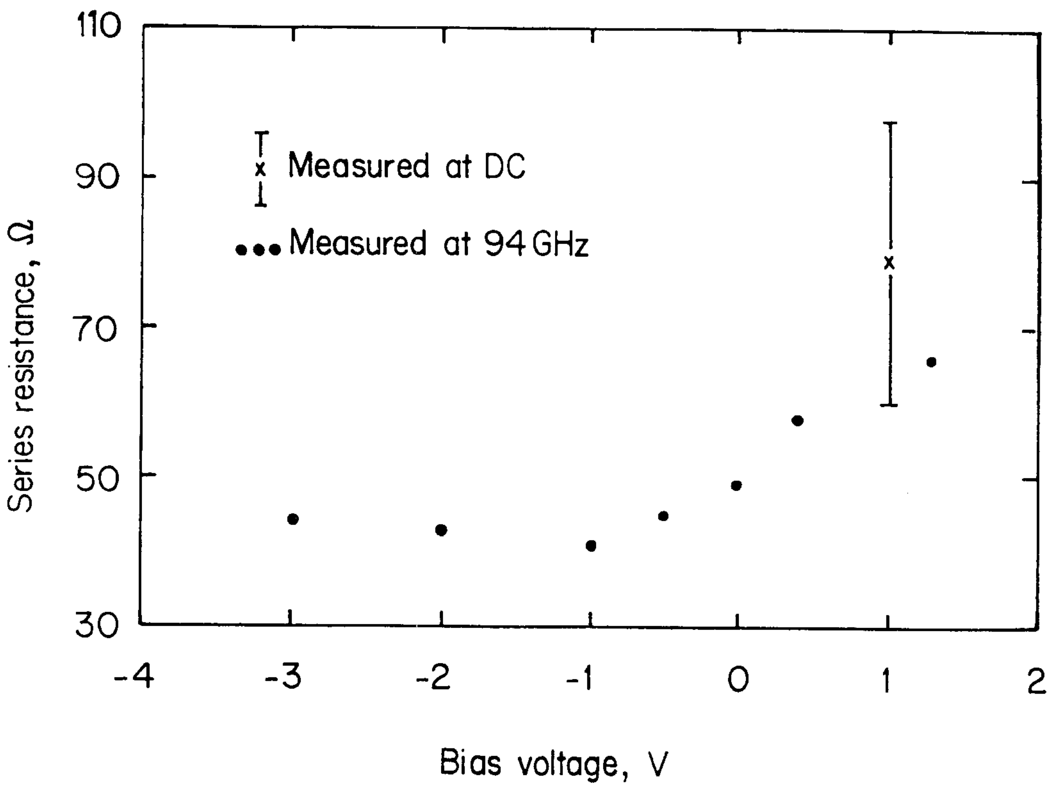


Figure 4.14b. A comparison between the measured resistance-voltage characteristic of the diode-grid at 94 GHz and the measured series resistance at DC. Bar represents one standard deviation.

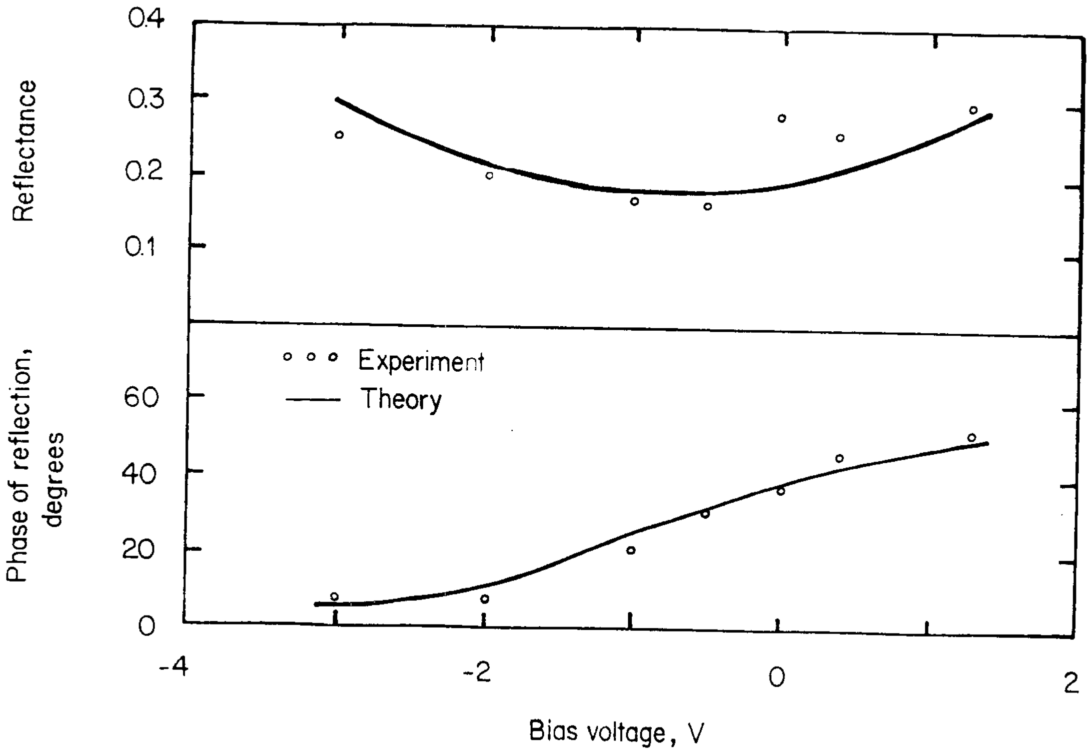


Figure 4.15. A comparison of the measured and the calculated phase shift performance as a function of bias voltage at 94 GHz.

In exploring the frequency dependence of the diode-grid, tuning curves of the same diode-grid were measured at several frequencies with zero bias on the grid. This was done in Professor Luhmann's laboratory at UCLA, where backward wave oscillators were available. Figure 4.16 shows the result of these measurements. The circles are the measured diode-grid impedance at frequencies as shown. The solid line shows the corresponding calculated diode-grid impedance, assuming a strip inductance of 0.26 nH, a diode series resistance of 78 Ω, and a zero bias diode capacitance of 30 fF. The agreement is reasonably good at 90 GHz, although it deteriorates quickly as the frequency approaches 130 GHz. This is expected since the effective dielectric wavelength is approaching the grid period (0.5 mm).

Figure 4.17 shows another configuration used in measuring phase shift performance of a diode-grid. A different diode-grid was used in this measurement. The largest phase shift occurred when the tuning mirror was placed 1.49 mm away from the diode-grid. A 70° phase shift and an average of 6.5-dB reflection loss was obtained. Figure 4.18 shows a comparison between the measured phase shift performance and calculation based on transmission-line theory. This comparison is complicated by the fact that the wafer thickness varies between 210 μm to 230 μm, 4 out of 35 rows of the diode-grid were shorted during the measurement, and the parasitic capacitance and series resistance of the diodes cannot be measured at low-frequency. The diode series resistance could not be measured accurately at DC because the surface of this wafer has many ripples with a feature size of about 1 μm. This created the problem of maintaining a good contact between the probes and the metal contacts; consequently, the contact resistance became too large (roughly 300 Ω) and dominated the actual diode series resistance. However, the diode series resistance was measured from a tuning curve at 93 GHz. The measured values were 26 Ω for the series resistance

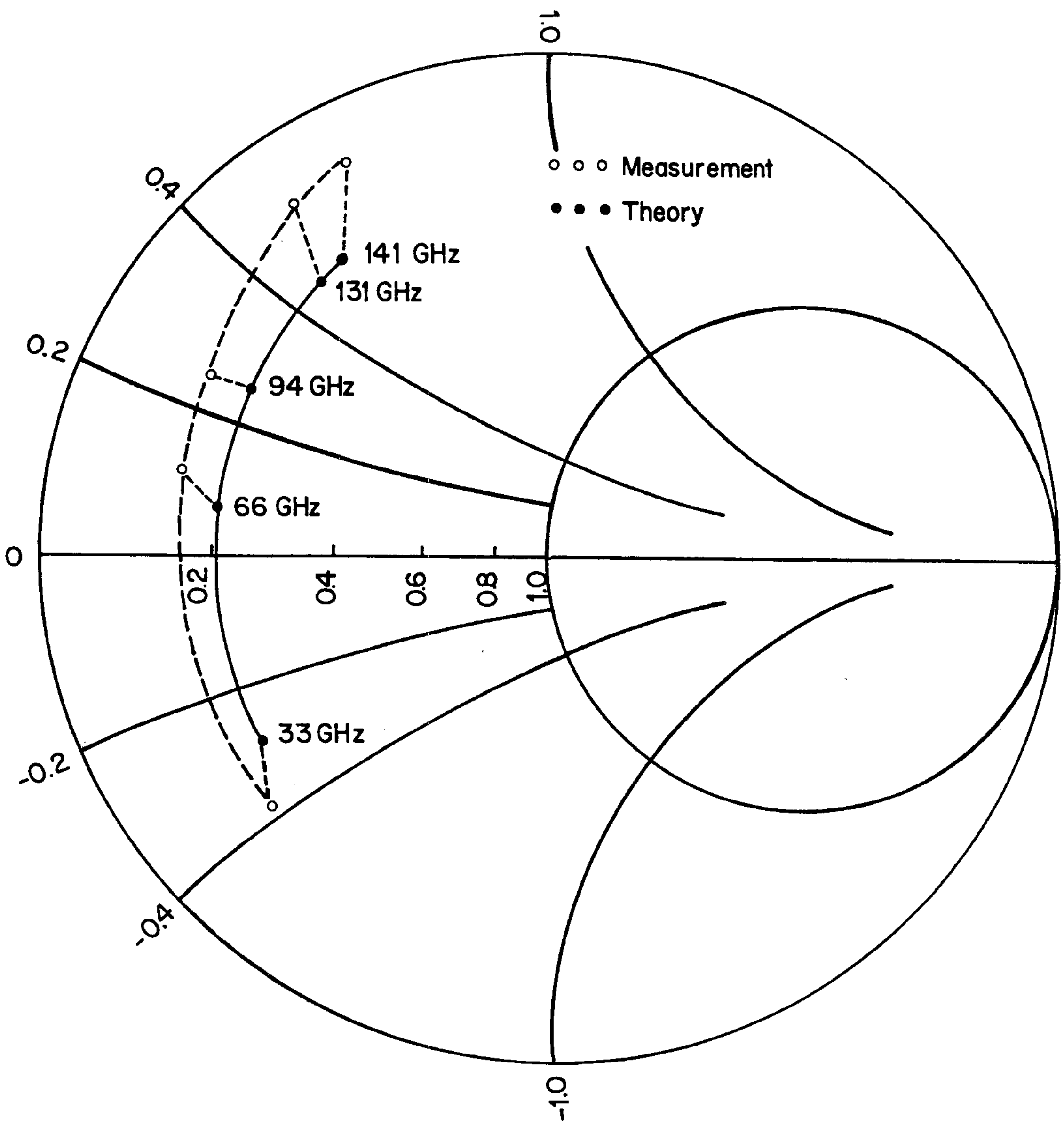
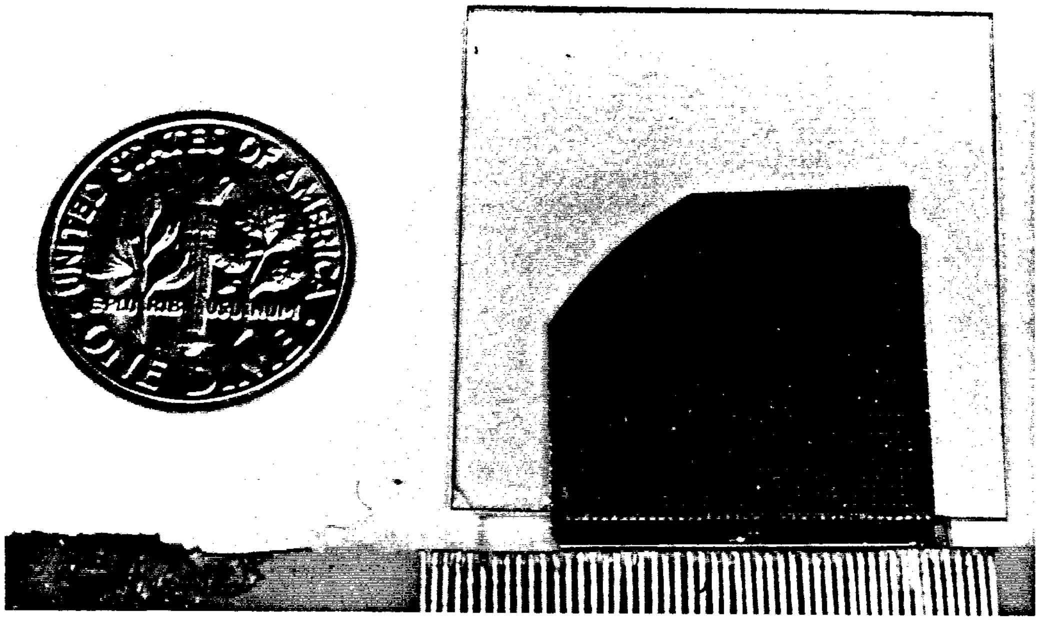
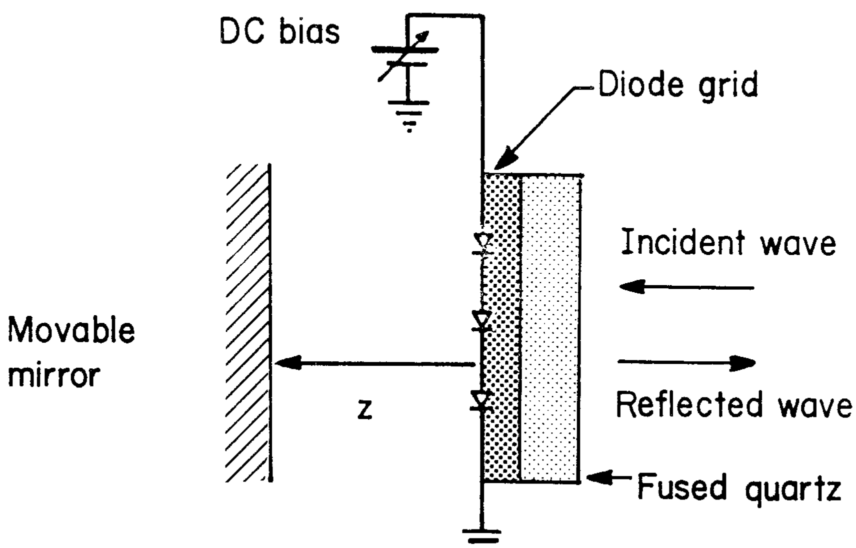


Figure 4.16. A comparison of the measured and the calculated diode-grid impedance as a function of the RF frequencies.



(a)



(b)

Figure 4.17. (a) Photograph of diode-grid on fused quartz. (b) Reflection measurement configuration of diode-grid on fused quartz.

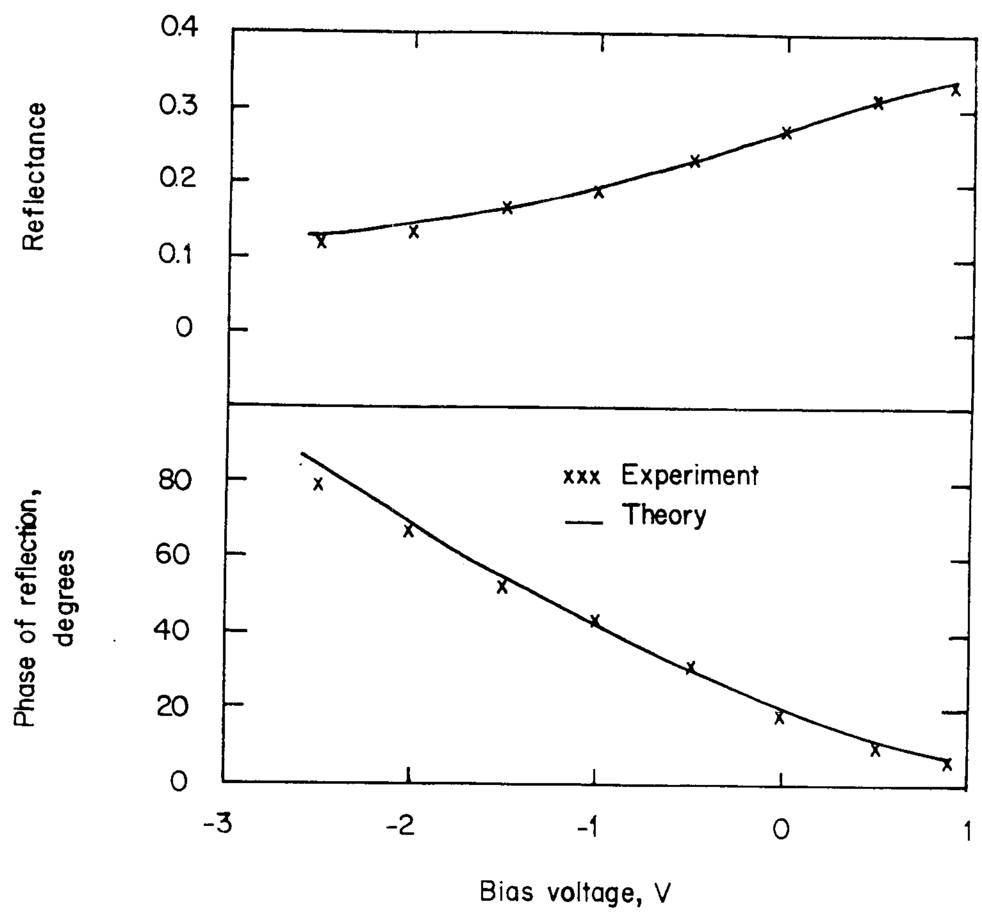


Figure 4.18. A comparison between the measured phase shift performance of a diode-grid in parallel with a fused quartz at 93 GHz.

and 62 Ω for the reactance. Theoretical curves were plotted using the measured series resistance (26 Ω), the average wafer thickness (218 μm), and the calculated grid inductive reactance (153 Ω). In addition, we assumed that the initial mirror position was 1.45 mm, the phase calibration length was 1 mm, the published refractive index for fused quartz and GaAs was 1.96 and 3.6, respectively [6], and the diode capacitance varied from 35 fF at 0.9 V to 18 fF at −2.75 V. The measured initial mirror position was 1.49 mm, the phase calibration length was 0.91 mm, and the average diode capacitance was 27 fF at zero bias and 20 fF at −3 V and had a standard deviation of 15 fF and 13 fF, respectively. Sensitivity analysis indicates that phase response is quite sensitive to wafer thickness and initial mirror position, shifting vertically 1° per micron for each.

References

[1] P. W. Hannan, "Simulation of a Phased-Array Antenna in Waveguide," *IEEE Trans. on Antennas and Propagation*, **AP-13,** pp. 342–353, 1965.

[2] R. G. Jones, "Precise Dielectric Measurements at 35 GHz Using an Open Microwave Resonator," *Proceedings IEE*, **Vol. 123,** no. 4, pp. 285–290, 1976.

[3] U. Stumper,"Dielectric Measurements by Multiport Reflectometers at Submillimeter Wavelengths," *11th Inter. Conf. on Infrared and Millimeter Waves*, Pisa, Italy, pp. 164–166, 1986.

[4] R. A. Bohlander, A. McSweeney, J. M. Newton, V. T. Brady, and R. G. Shackelford, "A Quasi-Optical Scanning Multiport (QUOSM) Network Analyzer," *6th Inter. Conf. on Infrared and Millimeter Waves*, F-2-5, 1981.

[5] N. W. B. Stone, J. E. Harris, D. W. E. Fuller, J. G. Edwards, A. E. Costley, J. Chamberlain, T. G. Blaney, J. R. Birch, and A. E. Bailey, "Electrical Standards of Measurement," *Proc. IEE*, **Vol. 122,** no. 10R, pp. 1054–1070, 1975.

[6] M. N. Afsar and K. J. Button, "Millimeter-Wave Dielectric Measurement of Materials" *Proc. of The IEEE*, **Vol.73,** no. 1, pp. 131–153, 1985.

[7] A. F. Harvey, "Optical Techniques at Microwave Frequencies," *Proceedings IEE*, **106B,** pp. 144–157, 1959.

[8] J. C. Wyant, "Interferometry for Three-Dimensional Sensing," *Test and Measurement World*, **April,** pp. 66-71, 1986.

[9] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, **Chap. 10,** pp. 274–334, Cambridge University Press, New York, 1986.

Chapter 5

Discussion and Future Work

In this thesis, several possible applications of diode-grids were proposed. In laying down the groundwork for these applications, a model of the diode-grid was presented and subsequently used in designing a diode-grid phaser shifter. A computer-aided design tool has been developed to provide an interactive graphics environment for doing designs and to form a basis for comparing theoretical and experimental results. A small aperture reflectometer that uses a wave-front interference technique has been developed to measure the reflection coefficient of diode-grids. The measured results have shown the diode-grid model to be reasonably accurate for doing the designs. A fabrication procedure for diode-grids has been demonstrated. The combination of Zah's self-aligning process and the liquid-crystal detection technique made it feasible. Monolithic diode-grids with 2000 Schottky varactors have been fabricated on $6\,\mathrm{cm}^2$ GaAs substrate. A phase shift of 70° with a 6.5-dB loss was measured for a single diode-grid. Further improvements remain to be done.

Although the fabrication process had a reasonable yield, the quality of the diode was not as good. The diode series resistance was too high. The breakdown voltage was too low; therefore, the ratio of the capacitance at zero bias to the capacitance at breakdown was too low. The average series resistance was $26\,\Omega$ for one grid and $78\,\Omega$ the other grid. The breakdown voltage for both diode-grids was about 3 V. This led to a capacitance ratio of about 2. Presently these limit the performance of the diode-grid.

One of the dominant factors of diode series resistance is the product of doping concentration and thickness of the n^+-layer. For the diode-grid that has a series resistance of $78\,\Omega$, the doping concentration and thickness of the n^+-layer were about $1\times10^{18}\,\mathrm{cm}^{-3}$ and $1.8\,\mu\mathrm{m}$, respectively. However, the diode series resistance can be lowered by increasing the doping concentration and thickness of n^+-layer. Ballamy and Cho [1] demonstrated beam-leaded diodes with series resistances of 4 to $8\,\Omega$. They used $1\times10^{18}\,\mathrm{cm}^{-3}$ for the doping concentration and $6\,\mu\mathrm{m}$ for the thickness of the n^+-layer. Clifton *et al.* [2] fabricated a Schottky diode that had a series resistance of $7\,\Omega$. They used $3\times10^{18}\,\mathrm{cm}^{-3}$ for the doping concentration and $3\,\mu\mathrm{m}$ for the thickness of the n^+-layer. Jarry *et al.* [3] developed a mixer diode with an incredibly low series resistance that was less than $3\,\Omega$. They used a doping concentration that was greater than $2\times10^{18}\,\mathrm{cm}^{-3}$ and 2 to $3\,\mu\mathrm{m}$ for thickness of the n^+-layer. These results indicate that there is an excellent chance for improving the series resistance of our diodes.

Among the factors that influence the diode breakdown voltage, contamination due to oil vapor back-streaming from the diffusion pump into the evaporating chamber was the most probable cause of our low diode breakdown voltage. The reverse breakdown characteristic was soft, and this led to a breakdown voltage of about 3 V at $0.5\,\mu\mathrm{A}\mu\mathrm{m}^{-2}$. This is about 25 % of Sze's prediction, which is 12 V for avalanche breakdown in a one-sided abrupt junction diode with a doping concentration of $1.5\times10^{17}\,\mathrm{cm}^{-3}$ [4]. The doping concentration for our diodes was $1.5\times10^{17}\,\mathrm{cm}^{-3}$ at the GaAs surface and decreased inversely as a function of the depth in the GaAs. The low breakdown voltage can be increased by using *in-situ* MBE aluminum, or by evaporating the metal for the Schottky contact in an oil-free ultra high vacuum system. For example, Cho and Dernier [5] deposited *in-situ* MBE aluminum on the GaAs epitaxy in the MBE growth chamber after the epitaxy was grown and before it was exposed to air. They fabricated Schottky diodes with 15 V breakdown voltage, which was about 60 % of Sze's prediction for diodes with a doping concentration of $5\times10^{16}\,\mathrm{cm}^{-3}$. Sato *el al.* [6] demonstrated that Schottky diodes fabricated in an oil vapor free vacuum had a higher breakdown voltage than diodes fabricated in an oil diffusion pumped vacuum. They indicated an 3.5 V to 5.3 V improvement for their diodes, which had a doping concentration of $3.5 \times 10^{17}\,\mathrm{cm}^{-3}$. The 5.3 V breakdown voltage corresponds to 75 % of Sze's prediction. In addition, Schottky diodes with near theoretical breakdown voltage have been fabricated by other researchers. Clifton *el al.* [2] fabricated diodes with 10 V breakdown voltage. This was about 80 % of Sze's prediction. Immorlica and Wood [7] developed diodes with 13 V breakdown voltage, and this was about 85 % of Sze's prediction. These were abrupt junction Schottky diodes. Their doping concentrations were $1.5 \times 10^{17}\,\mathrm{cm}^{-3}$ and $1 \times 10^{17}\,\mathrm{cm}^{-3}$, respectively. Their diode geometries were strips, which were similar to our diodes. Furthermore, their diode areas were about 10 μm, and proton bombardments were used for isolating their diodes. Although their methods of deposition were not given, these results do indicate that there is a good chance for improving the breakdown voltage and therefore the capacitance variation of our diodes.

Based on the reasonable agreement between theory and experiment, we believe our diode-grid model to be sufficiently accurate for doing the designs. Furthermore, works on second harmonic generation using diode-grids are being investigated by Christina Jou in Professor Luhmann's group at UCLA. They also showed a reasonable agreement between theory and experiment. For the diode-grid that had a series resistance of 78 Ω, the measured transmittance as a function of the position of the tuning slabs agreed well with the transmission-line model. Also, a second harmonic conversion efficiency of 16 % and an output power of 0.5 W were measured at 66 GHz when a pulsed magnetron at 33 GHz was used to pump the diode-grid that had a series resistance of 26 Ω.

These results indicate an exciting future for the diode-grid. In electronic beam-steering, the array design that is based on using two diode-grids appears to be feasible. Diode-grids for harmonic power generation look promising. Also, the integration of other electronic devices into a periodic grid is beginning to emerge. In our group, Zorana Popović is building a Gunn diode-grid on a Duroid substrate at 10 GHz. In any case, the future of integrating electronic devices into a periodic grid will be very exciting as well as promising.

References

[1] W. C. Ballamy and A. Y. Cho, "Planar Isolated GaAs Devices Produced by Molecular Beam Epitaxy," *IEEE Trans. on Electron Devices*, **ED-23**, pp. 481–484, 1976.

[2] B. J. Clifton, G. D. Alley, R. A. Murphy, and I. H. Mroczkowski, "High Performance Quasi-Optical GaAs Monolithic Mixer at 110 GHz," *IEEE Trans. on Electron Devices*, **ED-28**, pp. 155–157, 1981.

[3] B. Jarry, J. S. K. Mills, and F. Azan, "94 GHz Microstrip Monolithic GaAs Mixer," *IEE Electronics Letters*, **Vol. 22**, pp. 1328–1329, 1986.

[4] S. M. Sze, "P-N Junction Diode," *Physics of Semiconductors*, **Chap. 2**, pp. 63–132, John Wiley & Sons, Inc., 1982.

[5] A. Y. Cho and P. D. Dernier, "Single-crystal-aluminum Schottky-barrier diodes prepared by molecular-beam-epitaxy (MBE) on GaAs," *J. of Appl. Phys.*, **Vol. 49**, pp. 3328–3332, 1978.

[6] Y. Sato, M. Uchida, K. Shimada, M. Ida, and T. Imai, "GaAs Schottky Barrier Diode, ECL-1314," *Rev. of the Electrical Communication Lab.*, **Vol. 18**, pp. 638–644, 1970.

[7] A. A. Immorlica and E. J. Wood, "A Novel Technology for Fabrication of Beam-Leaded GaAs Schottky-Barrier Mixer Diodes," *IEEE Trans. on Electron Devices*, **ED-25**, pp.710–713, 1978.

Appendix A

Varactor Diode-Grid Fabrication Procedure

The following contains notes on the fabrication of a Schottky-barrier varactor diode-grid on a semi-insulating GaAs wafer. A total of five masks is used. Because a GaAs wafer is quite large and fragile, a holder made of teflon is used to hold the wafer during rinsing and developing. Figure A.1 shows the teflon holder. A teflon tweezer is used for etching. This process evolved from Zah's process [1]. The book by Ralph Williams is an excellent reference on GaAs processing techniques [2]. Howard Chen in Professor Yariv's group and Dr. Kjell Stolt in TRW have been the principal suppliers of the MBE wafers.

Fabrication Procedure

1. Obtain a GaAs wafer with the following layers.

   0.2 μm of *in situ* MBE aluminum.

   0.7 μm of n layer with a hyperabrupt doping profile.

   1.8 μm of n^+ layer with 3×10^{18} cm^{-3} doping concentration.

   * Some MBE wafers have indium on the back side and some do not. If indium is present, mount the wafer on a lapping block with wax and lap away the indium on the backside of the wafer. A mixture of water and a 5 μm diameter Al_2O_3 lapping powder made by Buehler is often used. Afterwards, acetone can be used to dissolve the wax.

2. Determine the crystal orientation of the wafer by making a strip pattern on a small chip scribed from the wafer. Note: If aluminum is present on the chip, remove the aluminum by etching it in aluminum etchant for GaAs.

   a. Standard lift-off photoresist process

   photoresist : AZ 1350J

spin speed : 4000 rpm

prebake : 85 °C for 25 min.

exposure : 25 sec.

development : 30 sec. (agitate in 1:1 diluted developer.)

b. Hardbake the photoresist at 125° C for 10 min.

c. Using a 93% concentrated H_2SO_4, and a 30% concetrated H_2O_2, mix the solution (H_2SO_4:H_2O_2:H_2O) with a ratio of 1:8:160. Stir it with a magnetic stirer in a petri-dish for an hour. The etch rate is about a quarter of a micron per min.

d. Etch the wafer and rinse it in 20 beakers of DI water.

e. Cleave a slice from the chip and mount it sideways on double-sided tape. Note the etched profile in a microscope. The diode orientation should be in the direction of a 'V-groove' etch or perpendicular to the direction of a 'dovetail' etch. Figure A.2 shows the etched profile.

3. If *in-situ* aluminum is not available, then clean the wafer throughly and evaporate aluminum.

   a. Cleaning procedure.

   acetone ultrasonic bath : 10 min.

   ethanol ultrasonic bath : 5 min.

   hot Transene 100 bath : 5 min.

   cold Transene 100 bath : 30 sec.

   let it dry by itself

   b. Etch away the native oxide on the wafer for a minute in a solution of (H_2O:HCl) with a 1:1 ratio and rinse it with 20 beakers of DI water.

   c. Load the wafer into the vacuum immediately. Thermo-paste is used to mount the wafer on a glass slide, which is then mounted onto the sample holder. Evaporate 2000 Å thick of aluminum in a vacuum with pressure below 3×10^{-6} Torr. If possible, this should be done in an oil-free vacuum system at lower pressure [4,5]. The aluminum source must be cleaned by etching it in organic solvent and aluminum etchant.

4. Generate the self-aligning mask for defining the ohmic contact and the width of the Schottky contact.

   a. Use the standard lift-off photoresist process. (see 2a.)

   b. Hardbake photoresist at 125° C for 10 min.

   c. Etch the aluminum in Transene aluminum etchant - Type D. Typically, it takes about 60 sec. at 58° C.

   d. Rinse the wafer in DI water and inspect it under the microscope. Look for signs of under-etched diode tips. They tend to form short ciruits when AuGe/Ni/Au is evaporated. Typically, several rounds of etching aluminum for 10s and rinsing the wafer are required to get good results. Because the wafer is quite large, some etching nonuniformity is expected. If the situation appears desperate, dissolve the photoresist and start over again since there are "tons" of aluminum still un-etched on the wafer. This is worth the trouble because an MBE wafer is precious.

5. Etch away the n-layer until the n^+-layer is exposed.

   a. Mix the solution (H_2SO_4:H_2O_2:H_2O) with a ratio of 1:8:160. Stir it with a magnetic stirrer in a Petridish for an hour. Note: Use the 93 % concentrated H_2SO_4, and the 30 % concetrated H_2O_2.

b. Etch the wafer and rinse it in 20 beakers of DI water. The etch rate is a quarter of a micron per min. Generally, an extra 15 second is added to over-etch the crystal. This is just a precautionary measure for exposing thoroughly the n^+ layer throughout the wafer.

6. Evaporate AuGe/Ni/Au with the following thicknesses (700Å/300Å/2000Å) at a pressure below 3×10^{-6} torr. The edges of the wafer are taped with

paper so that lift-off process in acetone becomes easier.

7. Remove the thermo-paste on the backside of the wafer with a Q-tip, that is slightly dampened with acetone. Lift off the photoresist in acetone and ethanol baths.

8. Generate the etching mask for defining the lead of the Schottky contact.

   a. Use the standard lift-off photoresist process. (see 2a.)

   b. Hardbake the photoresist at 125 °C for 10 min.

   c. Etch the aluminum (see 4 b.)

   d. Lift off the photoresist in acetone and ethanol baths.

9. Generate the bonding pad mask with standard lift off photoresist process. Evaporate AuGe/Ni/Au with the following thicknesses (700Å/300Å/2000Å) at pressure below 3×10^{-6} torr. The purpose for this is to make a good bonding pad. The alloying process make the metals partly dissolve into the GaAs and acquire a rough surface texture. This is worthwhile since wire bonding can be very difficult if the bond wires do not like to stick on the bonding pads.

10. Alloy the ohmic contacts and bonding pads at 460 °C for 10 min. in forming gas (15 % of N_2 and 85 % of H_2).

11. Generate proton implantation mask with photoresist.

    a. Pattern implantation mask.

       photoresist : AZ 4620

       spin speed : 4000 rpm

       prebake : 85 °C for 1 hour

       exposure : 70 sec. at 20 mJ-cm^{-2}

       development : 4 min. (agitate in 1:1 diluted developer)

    b. Optional: Some people feel that if the photoresist is flood-exposed here, then it will make the removal of the photoresist easier after ion implantation. This was not noticeable to me.

c. If implantation is to be done at Caltech, obtain sample holder from Frank So or Sung Kim in Professor Nicolet's group. Use the phosphorus compound to define the implantation boundary. Mount the wafer onto the sample holder with thermo-paste. If implantation is to be done at Hughes, no sample holder is required.

d. Implantation parameters:

330 keV at 4×10^{14} cm^{-2} dose

200 keV at 4×10^{12} cm^{-2} dose

12. Measure the resistance between the adjacent ohmic contacts to make sure that the desired area of the wafer is completely isolated. If isolation is completed, remove the photoresist in an O_2 plasma. Measure the diode parameters.
13. If the breakdown voltage varies widely across the wafer, then probing every varactor to weed out the weak diodes becomes necessary. This can be done at UCLA, since an analytical probing station is available for fast probing.
14. Measure the diode parameters with the HP4145B parameter analyzer and the CV characteristics with the HP4280A C-meter. See software documentation in the appendix. Here probing is quite tricky because contact resistance between the probe tip and the metal depends highly on the amount of pressure applied. Some practice is necessary to get a reasonably low contact resistance.
15. Generate the periodic grid mask and the bonding pad mask simultaneously with the standard lift-off photoresist process. Evaporate 100 Å of chrome and 3000 Å of gold. The lift-off here is critical. Lift-off flags must be avoided because they tend to cause electrical shorts. The cause is due to poor edge definition when the photoresist is patterned.

15\. Lap the wafer to the desired thickness.

16\. If a row of diode-grid is shorted, use liquid crystals to find shorted diodes and remove them from the wafer with an ultrasonic probe. It is important to prepare the sample properly. A layer of liquid crystal is spun onto the wafer at 1000 rpm. It is important to shake the bottle throughly before using it. This usually gives a nice and uniform layer. Use a Q-tip to wipe off the excess liquids on the bonding pad. Mount the wafer on a resistive heated chuck. Illuminate the wafer with a dual fiber optic lamp at approximately 30° incidence relative to the horizon. Use a curve tracer to inject current into the shorted row. Typically more than 5 mA is required to see any noticeable color change. Use the Signatone 850 ultrasonic cutter to remove the bad diode.

References

[1] C. E. Zah, "Fabrication Process for Monolithic Schottky Diode Imaging Arrays," *Ph. D. Thesis on Millimeter- Wave Monolithic Schottky Diode Imaging Arrays*, Chap. 3, pp. 103–109, California Institute of Technology, Pasadena, California, 1986.

[2] R. E. Williams, *Gallium Arsenide Processing Techniques*, Artech House, Inc., Massachusett, 1984.

[3] A. Y. Cho and P. D. Dernier, "Single-crystal-aluminum Schottky-barrier diodes prepared by molecular-beam-epitaxy (MBE) on GaAs," *J. of Appl. Phys.*, **Vol. 49,** pp. 3328-3332, 1978.

[4] Y. Sato, M. Uchida, K. Shimada, M. Ida, and T. Imai, "GaAs Schottky Barrier Diode, ECL-1314," *Rev. of the Electrical Communication Lab.*, **Vol. 18,** pp. 638–644, 1970.

[5] M. Ida, M. Uchida, K. Shimada, K. Asai, and S. Ishida, "Fabrication Technology of Stable Schottky Barrier Gates for Gallium Arsenide MESFETS," *Solid-State Electronics*, **Vol. 24,** pp. 1099–1105, 1981.

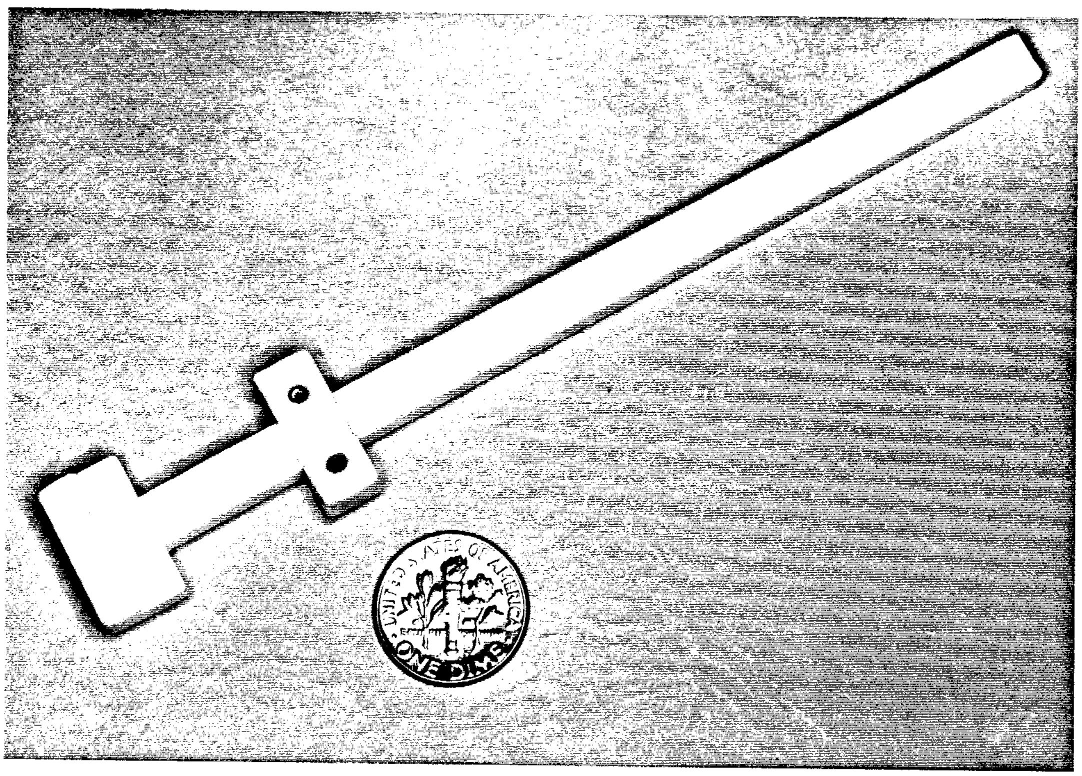
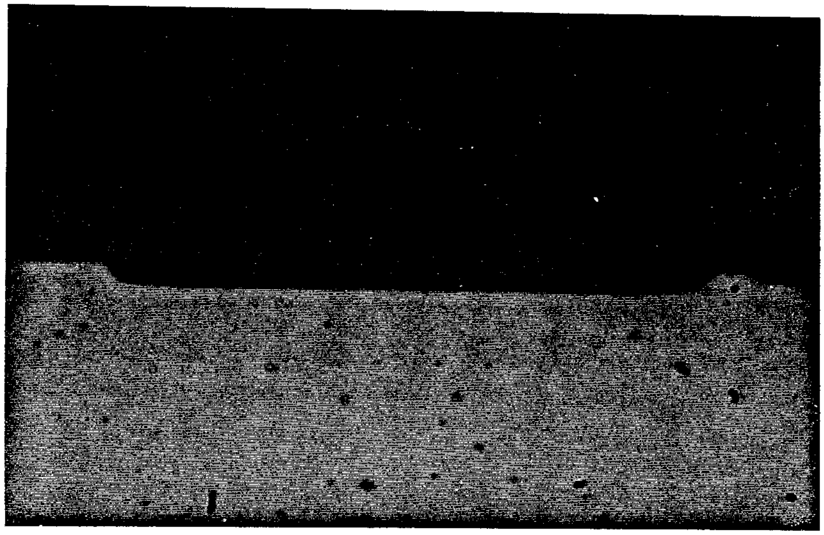
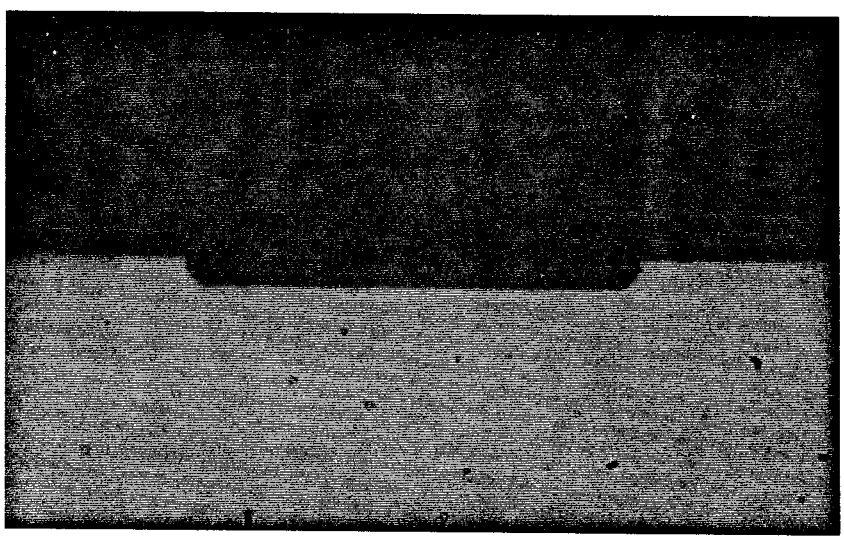


Figure A.1. A teflon holder for holding the GaAs wafer during developing and rinsing.



(a)



(b)

Figure A.2. Crystal orientation. (a) Cross-section profile of a "V-groove". (b) Cross-section profile of a "dove-tail".

Appendix B

Computer Program Listing of TRAP

```
program trap(input,output);


     { TRAP is an acronym that indicates the calculation of
     transmittance, reflectance, absorptance, and phase of
     reflection. Because the calculation of transmittance,
     absorptance and reflection coefficient for multi-layered
     media is tedious and time consuming, TRAP was developed to
     provide an interactive environment for the user to design the
     circuit and to compare the theoretical and experimental
     results. It is an interactive graphics program written in
     Turbo Pascal for an IBM-PC. For the computational algorithm,
     please see "Thin Films Calculations Using the IBM 650
     Electronic Calculator," by Jean A. Berning and
     Peter H. Berning in Journal of The Optical Society of America,
     Vol. 50, Num. 8, pg. 813, Aug. of 1960. TRAP features a
     line editor, from which the user can enter a command line
     that describes the structures of the layered media. Commands
     include lossy dielectric, grids, lumped elements,
     and a mirror. The angle of incidence, polarization,
     wavelength, and layer thicknesses can be varied linearly.
     Keyboard commands are available to stop, speed up,
     or slow down the simulation. TRAP also features optimization
     capability for the user to fit a transmission-line model
     to the measured reflectance and phase of reflection.
     The fitting procedure is based on minimizing the absolute
     value of the complex difference between the calculated and
     the measured reflection coefficient. For the optimization
     algorithm, please see Numerical Recipes by W. H. Press et al.,
     Chap. 10, pp. 274, Cambridge University Press, New York, 1986.

        When the program is run, a main menu is displayed. There
     are four options including database, simulation, optimization,
     graphics, and quit. To make a selection, simply press the key
     of the first letter for an option. For example pressing Q
     exits the program. Note also that pressing Q in an option exits
     that option, and pressing the return key repeats that option,
     although this is not explicitly displayed in each option.
     The database and the graphics option are menu driven and
     allow the user to read in a set of reflection coefficient
     data from an ASCII file and to define the vertical plotting
     range, respectively. The data file can be viewed or edited in
     the Turbo editor. The format of this file should be
     (distance   reflectance   phase of reflection). It should
     appear as three columns of numbers. Tyically these data are
     measured from an experiment, but they could be generated
     for the purpose of design and optimization. The simulation
     option allows the user to edit a command line describing
     the layered medium. The following are command definitions.
```

```

Convention : 1.) parameters, r1, r2, r3, ... etc., are real numbers.
             2.) i, b, f, ... etc.  are definitions.
             3.) , and :  are delimiters.
             4.) a command is usually followed by a set of parameters.

ir1             - incident medium : r1 = refractive index
                                    (default value is 1)
br1,r2          - dielectric boundary: r1 = Re(refractive index)
                                       r2 = Im(refractive index)
ar1,r2          - shunt admittance : r1 = Re(shunt admittance)
                                     r2 = Im(shunt admittance)
fr1,r2          - final medium : r1 = Re(refractive index)
                                 r2 = Im(refractive index)
sr1             - TE polarization : r1 = angle of incidence in
                                         degrees with respect to
                                         the surface normal.
pr1             - TM polarization : r1 = angle of incidence in
                                         degrees with respect to
                                         the surface normal.
wr1             - wavelength of incidence : r1 =  wavelength in
                                                  arbitrary units.
gr1,r2,r3,r4    - quasi-static model of a square grid :
                  r1 = length of the period.
                  r2 = length of the gap.
                  r3 = length of the post.
                  r4 = series resistance.
Gr1,r2,r3,r4    - Eisenhart model of a square grid :
                  r1 = length of the period.
                  r2 = length of the gap.
                  r3 = length of the post.
                  r4 = series resistance.
tr1             - layer thickness : r1 = wavelength in arb. units.
d               - plot y-axis in unit of dB. (default is linear)
xr1:r2          - plot the variable (x: i,b,a, ... etc. ) on x-axis.
                  r1 = start value of x
                  r2 = stop value of x

    Examples: 1.) i1 z0,.11 b3.6 t.233 z0,.11 f1 s0 w1.7:2.7
              2.) i1 b3.6 t.233 z.054,.27 b1 t0:2.0 m s0 w3.36
              3.) i1 b3.6 t.233 z.054,.1:.3 b1 t1.5 m s1 w3.36

   The first example is a bandpass filter formed by a pair of
lossless inductive screens. The second example is a zero-bias
diode-grid backed by a mirror that translates from 0 to 2.0 mm.
The third example is a diode-grid with varying reactance and
a stationary mirror tuned to 1.5 mm. These examples had been
programed. To run them, enter the simulation mode and press E.
The description of the multi-layered medium can be edited
using a condensed version of Wordstar commands:

    control s - left
    control d - right
```

```
          control g - delete
          control v - change from insert mode to overwrite mode and back.
          backspace - deletes left

         When the layer description is correct, press carriage
      return to enter it. You can get a screen dump by typing shift
      PrtSc. The optimization mode allows the user to fit a set
      of reflection coefficient data to a transmission-line model
      of the layered medium. The data can be read in from an ASCII file
      thru the database option. Once the data are entered, enter Q to
      return to main menu and press O to enter optimization option.
      The commands for optimization are similar to simulation, except
      that the command should be in capital letters to signify that
      the variable is to be optimized. Following the command symbol
      is the optimization range (r1;r2), where a semicolon is used
      to separate the minimum (r1) and the maximum (r2).

         Example: 4.) i1 b1 T.68;.69 Z.042;.043,.085;.09
                      b3.6 t.218 b1 T2.49;2.5 t0:2.0 m s0 w3.36

      The fourth example illustrates an optimization command line for
      curve-fitting a transmission-line model of a diode-grid with
      a tuning mirror to the experiment. To run this example, enter
      the database option, get the measured reflection coefficient
      from the file "diogrid.pas," return to main menu, enter the
      optimization mode, press the key E, enter 4, and hit return.
      The data file is available in the disc on the back cover of
      this thesis. It was obtained from an actual reflection
      measurement and is shown in part below. As the optimization
      advances, the values of each parameter of the multi-layered
      medium are displayed line by line on the screen. Each line
      represents a completed computation. The ordering of the
      parameters in the line starts from the right of the medium
      to left of the medium, and the associated error is displayed
      last in the line.

            0.000    0.912   24.704
            0.050    0.903   24.485
            0.100    0.886   24.534

                       .
                       .
                       .

            1.900    0.937   24.039
            1.950    0.944   23.760
            2.000    0.953   23.134        }



type
   line = string[250];
   complex = ^complex_record;
```

```
      complex_record = record
      r,i : real;
      end;
   complex_matrix = ^complex_matrix_record;
      complex_matrix_record = record
      t11,t12,t21,t22 : complex;
      end;
   char_set = set of char;
   glmpnp = array[1..9,1..8] of real;
   glmp = array[1..9] of real;
   glnp = array[1..8] of real;
   rng = array[1..2,1..8] of real;

const
   structure : line = '';
   structuresave : line = '';
   xmin = 190; xmax = 550; {coordinates of graph corners}
   ymin = 43; ymax = 163;
   xd = 3; {plotting interval on x_axis}
   backspace = #8; enter = #13;
   edit_set : set of char = [^d,^s,^g,^v, backspace];
   structure_set : set of char = ['-','.','0','1','2','3','4','5',
                                 '6','7','8','9',',',':',' ',
                                 'w','i','b','t','g','d','s','r','n','o',
                                 'z','a','p','f','m','j','q','G','/',';',
                                 'B','T','Z','A','I','F','W'];
   number_set:set of char=['-','.','0','1','2','3','4','5','6','7','8','9'];
   delimiter_set : set of char = [',',':','w','i','b','t','g','d','s','p',
                                 'o','z','a','f','m','j','q','G','/','r',
                                 'n',';','B','T','Z','A','I','F','W'];
   command_set : set of char = ['w','i','b','t','g','d','a','z','r','n',
                               'o','s','p','f','m','j','q','G','/',';',
                               'B','T','Z','A','I','F','W'];

label pau, main;

var
   position, pointer, positionstop : 1..150; {position in structure.}
   a_heap_pointer, b_heap_pointer : ^integer; {pointer for garbage disposal}
   x, i : integer; {position along the x-axis, 0 to xmax-xmin. }
   t : complex_matrix;    {complex transmittance matrix}
   reflectance, transmittance, absorptance, phase : real;
   error_code : integer; {for val statement}
   min, max : real; { minimum and maximum values of the x coordinates}
   c, b : char; {character read at keyboard}
   dB_plot : boolean; {dB plot if true}
   yj, rc, pjao, pj, yjm1, cnj, cnje, gj,
   zj, aj, cno, cnoe, yjm1n, yjm1d, caj, cnjp : complex;
   no, sj, ao, nj, l, lj, sjt : real;
   G_flag, p_flag : string[2];
   p, d, v, h, QQ, ss, ccc, f1, f2, XX, BB : real;
   a, g, w, rs, rd, cv, sjsave : real;
   scan : string[5];
```

```
    bc, bl, par : complex;
    m, n, en, positionsave, positionG, positionf : integer;
    zd, bcv, yjsave, cnosave, ymnp, ymnm, bcsum, blsum, densum : complex;
    xyz, xyz2 : complex;
    test_string : line;
    temp, la, mla, nla, ld, sinc : real;
    yjsaver, yjsavei, cnosaver, cnosavei : real;
    intensity, normalization : real;
    on_off : integer;
    select : char;
    filename : string[15];
    trp : array [1..100 , 1..3] of real;
    trp1, trp2, trp3 : real;
    data_transfer : text;
    data_index, ndata, data_inc : integer;
    find_error_flag : boolean;
    ref_err : real;
    xmi, ylmi, yrmi, xmx, ylmx, yrmx : real;
    ylrange, ylscale, yrrange, yrscale, xrange, xscale : real;
    zzzx, zzzy : integer;
    compon : array[1..20] of real;
    rtol, ftol : real;
    nparam, idata, optimization_flag, mvertices, ndim, iter : integer;
    poly : glmpnp;
    vert : glnp;
    face : real;
    y : glnp;
    range : rng;
    range_variable : complex;
    range_pointer, curve_select : integer;


{GRAPHICS routines}


procedure draw_box (x1,y1,x2,y2,color : integer);
begin
   draw (x1,y1,x1,y2,1);
   draw (x1,y2,x2,y2,1);
   draw (x2,y2,x2,y1,1);
   draw (x2,y1,x1,y1,1);
end;

procedure draw_x_ticks(x1,y1,x_increment,x2 : integer);
begin
   while (x1<=x2) do begin
   draw (x1,(y1+1),x1,(y1-2),1);
   x1 := x1 + x_increment;
   end; {while}
end; {draw_x_ticks}

procedure draw_y_ticks(x1,y1,y_increment,y2 : integer);
begin
```

```
   while (y1<=y2) do begin
   draw ((x1+2),y1,(x1-4),y1,1);
   y1 := y1 + y_increment;
   end; {while}
end; {draw_y_ticks}

procedure draw_graph(x1,y1,x2,y2,
   x_increment, y_increment_left, y_increment_right :integer);
begin
   draw_box(x1,y1-2,x2,y2+2,1);
   draw_x_ticks (x1,y1,x_increment,x2);
   draw_x_ticks (x1,y2,x_increment,x2);
   draw_y_ticks (x1,y1,y_increment_left,y2);
   draw_y_ticks (x2,y1,y_increment_right,y2);
end; {procedure draw_graph}

procedure write_x_coordinates(x1,y1,x_increment :integer;
   x1c,xc_increment,x2c : real; field,fix :integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (x1c:field:fix);
   x1c := x1c + xc_increment;
   x1 := x1 + x_increment;
   until x1c > x2c;
end; {write_x_coordinates}

procedure write_y_coordinates(x1,y1,y_increment :integer;
   y1c,yc_increment,y2c : real; field, fix : integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (y1c:field:fix);
   y1c := y1c + yc_increment;
   y1 := y1 + y_increment;
   until y1c < y2c;
end; {write_y_coordinates}

function dB(x : real) : real; {give the dB difference}
begin
dB := 10*ln(x)/ln(10);
end; {dB}

procedure clean_box;
begin
  gotoxy(2,2);
  write('                                                              ');
  gotoxy(2,3);
  write('                                                              ');
end;

procedure set_up;
begin
```

```
    if select = 'm' then begin
              hires; hirescolor(15);
              draw_box(0,4,639,33,1);
              gotoxy(2,2);
              write('MAIN MENU: ');
              write('s(simulation) d(database) ');
              write('o(optimization) g(graphics)');
              if c = 'q' then read(kbd,select);
              if (select = 's') or (select = 'd') then
                begin
                draw_graph(xmin,ymin,xmax,ymax,(xmax-xmin)div 5,
                (ymax-ymin)div 5, (ymax-ymin)div 4);
                if select = 's' then begin
                      gotoxy(1,7); write('Tran.  . . . . .');
                      gotoxy(1,8); write('Refl.  ..  ..  ..');
                      gotoxy(1,9); write('Abs.   ...   ...');
                      gotoxy(1,10); write('Pha.   .......');
                end;
              end;
    end;
    graphwindow(0,0,639,199);
    case select of
      's' : begin
              clean_box;
              gotoxy(2,2); write('Key: i(nt) b(ndry) t(kns) g(rd) ');
              writeln('f(nl) w(vlth) m(ir) G(*) s(TE) p(TM) d(B) q(it)');
              gotoxy(2,3); writeln('Structure:');
            end;
      'd' : begin
              clean_box;
              gotoxy(2,2); write('DATABASE: enter data_filename ');
            end;
      'o' : begin
              optimization_flag := 1;
              clean_box;
              gotoxy(2,2); write('Key: i(nt) b(ndry) t(kns) g(rd) ');
              writeln('f(nl) w(vlth) m(ir) G(*) s(TE) p(TM) d(B) q(it)');
              gotoxy(2,3); writeln('Structure:');
            end;
    end;  {of case}

end; {set_up}

procedure write_y_axis;
begin
   write_y_coordinates(xmax+20,ymin+7,(ymax-ymin) div 4,
                       yrmx,(yrmi-yrmx)/4,yrmi,4,0);
   if dB_plot then
      write_y_coordinates(xmin-20,ymin+7,(ymax-ymin) div 5,
                          0,-10,-50,3,0)
      else
      write_y_coordinates(xmin-20,ymin+7,(ymax-ymin) div 5,
                          ylmx,(ylmi-ylmx)/5,ylmi,3,1);
```

```
end; {write_y_axis}

procedure write_x_axis;
begin
   write_x_coordinates (xmin,ymax+14,(xmax-xmin) div 5,
                        min,(max-min)/5,max,4,1);
end; {write_x_axis}

procedure plot_result(on_or_off : integer);
begin
   graphwindow (xmin,ymin,xmax,ymax);
   trp3 := yrmx - phase*180.0/pi;
   if (on_off and $8) <> 0 then plot(x*xd,round(trp3*yrscale),on_or_off);
   if dB_plot then begin
      if odd(x) then
         if (on_off and $4) <> 0 then
            plot(x*xd,round(-dB(transmittance)/50*(ymax-ymin)),on_or_off);
      if odd(x div 2) then
         if (on_off and $2) <> 0 then
            plot(x*xd,round(-dB(reflectance)/50*(ymax-ymin)),on_or_off);
      if odd(x div 3) then
         if (on_off and $1) <> 0 then
            plot(x*xd,round(-dB(absorptance)/50*(ymax-ymin)),on_or_off);
   end  {if}
   else begin
      trp1 := ylmx - transmittance;
      trp2 := ylmx - reflectance;
      trp3 := ylmx - absorptance;
      if odd(x)  then
         if (on_off and $4) <> 0 then
            plot(x*xd,round(trp1*ylscale),on_or_off);
      if odd(x div 2) then
         if (on_off and $2) <> 0 then
            plot(x*xd,round(trp2*ylscale),on_or_off);
      if odd(x div 3) then
         if (on_off and $1) <> 0 then
            plot(x*xd,round(trp3*ylscale),on_or_off);
   end; {else}
end; {plot_result}


{COMPLEX NUMBERS routines}


function co (s,t : real) : complex; {makes a complex number}
var
   u : complex;
begin
   new(u);
   u^.r := s;
   u^.i := t;
   co := u;
end;
```

```
function sc(s : real; t : complex) : complex;
{multiply a real number s by t}
var
   u : complex;
begin
   new(u);
   u^.r := s * t^.r;
   u^.i := s * t^.i;
   sc := u;
end;

function cc(s : complex) : complex; {complex conj}
var
   u : complex;
begin
   new(u);
   u^.r := s^.r;
   u^.i := -s^.i;
   cc := u;
end;

function ma(s : complex) : real; {magnitude}
begin
   ma := sqrt(sqr(s^.r)+sqr(s^.i));
end;

function ph(s : complex) : real; {phase in radians, (-pi,+pi]}
begin
   if s^.r > 0 then ph := arctan(s^.i/s^.r);
   if s^.r < 0 then if s^.i >= 0 then ph := arctan(s^.i/s^.r) + pi
      else ph := arctan(s^.i/s^.r) - pi;
   if s^.r = 0 then begin
         if s^.i > 0 then ph := pi/2;
         if s^.i < 0 then ph := -pi/2;
         if s^.i = 0 then ph := 0;
      end; {real part 0}
end;

function su(s,t : complex) : complex; {sum}
var
   u : complex;
begin
   new(u);
   u^.r := s^.r + t^.r;
   u^.i := s^.i + t^.i;
   su := u;
end;

function pr(s,t : complex) : complex; {product}
var
   u : complex;
begin
```

```
   new(u);
   u^.r := s^.r*t^.r - s^.i*t^.i;
   u^.i := s^.r*t^.i + s^.i*t^.r;
   pr := u;
end;

function di(s,t : complex) : complex; {difference s minus t}
var
   u : complex;
begin
   new(u);
   u^.r := s^.r - t^.r;
   u^.i := s^.i - t^.i;
   di := u;
end;

function qu(s,t : complex) : complex; {quotient s over t}
var
   u : complex;
begin
   new(u);
   u^.r := (s^.r*t^.r + s^.i*t^.i)/(sqr(t^.r) + sqr(t^.i));
   u^.i := (-s^.r*t^.i + s^.i*t^.r)/(sqr(t^.r) + sqr(t^.i));
   qu := u;
end;

function sq(s : complex) : complex; {square}
var
   u : complex;
begin
   new(u);
   u^.r := s^.r*s^.r - s^.i*s^.i;
   u^.i := 2*s^.r*s^.i;
   sq := u;
end;

function rr(s : complex) : complex;
{square root in the right half plane.}
var
   u : complex;
begin
   new(u);
   u^.r := sqrt(ma(s))*cos(ph(s)/2);
   u^.i := sqrt(ma(s))*sin(ph(s)/2);
   rr := u;
end;

function ur(s : complex) : complex;
{square root in the upper half plane.}
var
   u : complex;
   theta : real;
begin
```

```
   new(u);
   if ph(s) >= 0 then theta := ph(s)/2 else theta := ph(s)/2 + pi;
   u^.r := sqrt(ma(s))*cos(theta);
   u^.i := sqrt(ma(s))*sin(theta);
   ur := u;
end;

function lr(s : complex) : complex;
{the square root in the lower half plane.}
var
   u : complex;
   theta : real;
begin
   new(u);
   if ph(s) <= 0 then theta := ph(s)/2 else theta := ph(s)/2 - pi;
   u^.r := sqrt(ma(s))*cos(theta);
   u^.i := sqrt(ma(s))*sin(theta);
   lr := u;
end;

function ex(s : complex) : complex; {exponential function}
var
   u : complex;
begin
   new(u);
   u^.r := exp(s^.r)*cos(s^.i);
   u^.i := exp(s^.r)*sin(s^.i);
   ex := u;
end;

function sinh(s : complex) : complex; {hyperbolic sine}
var
   u : complex;
begin
   new(u);
   u^.r := cos(s^.i)*(exp(s^.r)-exp(-s^.r))/2;
   u^.i := sin(s^.i)*(exp(s^.r)+exp(-s^.r))/2;
   sinh := u;
end;

function cosh(s : complex) : complex; {hyperbolic sine}
var
   u : complex;
begin
   new(u);
   u^.r := cos(s^.i)*(exp(s^.i)+exp(-s^.r))/2;
   u^.i := sin(s^.i)*(exp(s^.r)-exp(-s^.r))/2;
   cosh := u;
end;


function sine(s : complex) : complex; {sine function}
var
```

```
   u : complex;
begin
   new(u);
   u^.r := sin(s^.r)*(exp(s^.i)+exp(-s^.i))/2;
   u^.i := cos(s^.r)*(exp(s^.i)-exp(-s^.i))/2;
   sine := u;
end;

function cosine(s : complex) : complex; {cosine function}
var
   u : complex;
begin
   new(u);
   u^.r := cos(s^.r)*(exp(s^.i)+exp(-s^.i))/2;
   u^.i := sin(s^.r)*(-exp(s^.i)+exp(-s^.i))/2;
   cosine := u;
end;

function cm(a11,a12,a21,a22:complex):complex_matrix;
{makes a complex matrix}

var
   u : complex_matrix;

   function eq (s : complex) : complex; {makes an equal complex number.}
   var
      u : complex;
   begin
      new(u);
      u^.r := s^.r; u^.i := s^.i;
      eq := u;
   end; {eq}

begin
   new(u);
   with u^ do begin
      t11 := eq(a11); t12 := eq(a12);
      t21 := eq(a21); t22 := eq(a22);
      end; {with}
   cm := u;
end; {cm}

function mp (a,b : complex_matrix) : complex_matrix;
{complex matrix product}
var
   u : complex_matrix;
begin
   new(u);
   with u^ do begin
      t11 := su(pr(a^.t11,b^.t11),pr(a^.t12,b^.t21));
      t12 := su(pr(a^.t11,b^.t12),pr(a^.t12,b^.t22));
      t21 := su(pr(a^.t21,b^.t11),pr(a^.t22,b^.t21));
      t22 := su(pr(a^.t21,b^.t12),pr(a^.t22,b^.t22));
```

```
   end; {with}
   mp := u;
end; {function mp}


{LINE_EDITOR routines}


function edit(x,y : integer; test_string : line) : line;

var
   ins: boolean; {true for insert mode on}
   position : 1..150;
   row, col,px,py : integer;

label quit;

procedure write_cursor_position;

begin
   gotoxy(row,col);
   write('   ');
   gotoxy(row,col);
   write(position);
end; {write_cursor_position}

begin
   ins:= true; {insert mode on}
   position:= 1;
   row := 72; col := 4;
   write_cursor_position;
   gotoxy(x,y); write(test_string);
   repeat
     {draw cursor}
     draw(8*(x+position-1)-8,8*y-1,8*(x+position-1)-1,8*y-1,1);
     draw(8*(x+position-1)-8,8*y-2,8*(x+position-1)-1,8*y-2,1);
     gotoxy(x+position-1,y);
     read(kbd,c);
     {erase cursor}
     draw(8*(x+position-1)-8,8*y-1,8*(x+position-1)-1,8*y-1,0);
     draw(8*(x+position-1)-8,8*y-2,8*(x+position-1)-1,8*y-2,0);
     if c in structure_set then begin
         insert(c,test_string,position);
         position:= position+1;
         write_cursor_position;
         if not(ins) then delete(test_string,position,1);
      end;
      if c in edit_set then begin
         case c of
            ^d : if position <= ord(test_string[0]) then begin
                   position:= position+1;
                   write_cursor_position;
                   end;
```

```
          ^g : begin
                 delete(test_string,position,1);
                 gotoxy(x+ord(test_string[0]),y);
                 write(' ');
                 end;
          ^v : ins:= not(ins);
          ^s : if position > 1 then begin
                 position:= position-1;
                 write_cursor_position;
                 end;
          backspace : if position > 1 then begin
                        position:= position-1;
                        write_cursor_position;
                        delete(test_string,position,1);
                        gotoxy(x+ord(test_string[0]),y);
                        write(' ');
                        end;
       end; {case}
    end;
    if c = 'e' then begin
    gotoxy(x,y);
    write('There are 3 examples. Enter 0 -> quit & 1,2 or 3 -> exmpl: ');
    read(kbd,c);
    case c of
       '1' : test_string := 'i1 z0,.11 b3.6 t.233 z0,.11 f1 s0 ' +
                            'w1.7:2.7';
       '2' : test_string := 'i1 b3.6 t.233 z.054,.27 b1 t0:2.0 ' +
                            'm s0 w3.36';
       '3' : test_string := 'i1 b3.6 t.233 z.054,.1:.3 b1 t1.5 ' +
                            'm s1 w3.36';
       '4' : test_string := 'i1 b1 T.68;.69 Z.042;.043,.085;.09 ' +
                            'b3.6 t.218 b1 T2.49;2.5 b1 t0:2 m ' +
                            's0 w3.36';
       '5' : test_string := 'i1 b1 T.001;.2 b1.96 t.434 b3.6 t.231 ' +
                            'Z.045;.065,.25;.31 b1 T.6;0.8 b1 t0:2.0 ' +
                            'm s0 w3.36 o10';
       '6' : test_string := 'i1 b1.96 t.59 b1 t0:2.3 b1.96 t.59 b1 ' +
                            't0:2.3 b3.23 t6.72 b1 t2.3 b3.6 t.238 ' +
                            'z.16,-.13 b1 t1.15 m s0 w4.6';
       '7' : test_string := 'i1 b1.96 t.3 b1 t0:1.2 b1.96 t.3 b1 ' +
                            't0:1.2 b3.23 t6.72 b1 t2.3 z.27,.29 ' +
                            'b3.6 t.238 b1 t.86 m s0 w2.3';
    end; {case}
    gotoxy(x,y);
    writeln('                                                             ');
    if (c = '1') or (c = '2') or (c = '3') or
       (c = '4') or (c = '5') or (c = '6') or
       (c = '7') then c := enter;
    end; {if}
    gotoxy(x,y);
    write(test_string);
 until c in [enter, 'q'];
 edit:= test_string;
```

```
quit:
end; {edit}


{CALCULATION rountines}


function rp : real; {finds a real parameter  in structure description}

   begin
      rp := 0;
      test_string := '';
      pointer := pointer + 1;
      while not((pointer > ord(structure[0])) or
            (structure[pointer] in delimiter_set)) do begin
         if structure[pointer] in number_set then
            test_string := test_string + structure[pointer];
         pointer:= pointer + 1;
      end; {while}
      if test_string <> '' then begin
         val(test_string,temp,error_code);
         rp := temp;
         end; {if}
      if (pointer < ord(structure[0])) and (structure[pointer] = ':')
         then begin
         min := temp;
         max := rp;
         if optimization_flag=1 then
         rp := trp[idata,1]
         else
         rp := min + (max-min)*x*xd/(xmax-xmin);
      end; {if}
   end; {rp}

function cp : complex;
{finds a complex parameter in structure description}
   var
      u : complex;
   begin
      new(u);
      u^.r := rp;
      if (structure[pointer] = ',') or (structure[pointer] = ';') then
      begin
         u^.i := rp;
      end {if}
      else u^.i := 0;
      cp := u;
   end;  {cp}

function orp : real; {returns a real value for optimization}
   begin
      nparam := nparam + 1;
      orp := vert[nparam];
```

```
   end;

function ocp : complex; {returns a complex value for optimization}
   begin
      nparam := nparam + 1;
      ocp^.r := vert[nparam];
      nparam := nparam + 1;
      ocp^.i := vert[nparam];
   end;

function ymnj : complex;
   begin
      la := (l*l)/(nj*nj*a*a);
      mla := m*m*la;
      nla := n*n*la;
      ymnj := qu(co((1-mla)*nj,0),lr(di(co(1,0),su(co(mla,0),co(nla,0)))));
   end; {equivalent characteristic admittance}

procedure incident_medium;
      begin
      if structure[position] = 'i' then cno := cc(cp)
      else cno := ocp;
      if p_flag = 'TE' then cnoe := sc(cos(ao),cno);
      if p_flag = 'TM' then cnoe := sc(1/cos(ao),cno);
      {if G_flag='of' then
      write('i',round(cno^.r),',',round(-cno^.i),' ');}
   end; {incident_medium}

procedure neff;
      begin
      caj := lr(di(co(1,0),sc(sin(ao)*sin(ao),sq(qu(cno,cnj)))));
      if p_flag = 'TE' then begin
      cnje := pr(cnj,caj); write('');
      end;
      if p_flag = 'TM' then begin
      cnje := qu(cnj,caj); write('');
      end;
   end; {effective refractive index}

procedure transmit_admittance;
   begin
      if G_flag = 'on'
         then
           begin
             ld := l/nj;
             mla := m*m*ld*ld/(a*a);
             nla := n*n*ld*ld/(a*a);
             pj := sc(2*pi*lj/ld,lr(co(1-mla-nla,0)));
             if pj^.i <= -200 then pj^.i := -200;
           end
         else
           begin
             pj := sc(((2*pi)/l)*lj,cnj);
```

```
            end;
      pjao := pr(pj,caj);
      yjm1n := su(pr(yj,cosine(pjao)),pr(pr(co(0,1),cnje),sine(pjao)));
      yjm1d := su(cosine(pjao),qu(pr(pr(co(0,1),yj),sine(pjao)),cnje));
      yjm1 := qu(yjm1n,yjm1d);
   end; {transmit_admittance}

procedure loss;
   begin
      if yjm1^.r <>0 then
      begin
      sjt := yj^.r/(yjm1^.r*ma(yjm1d)*ma(yjm1d));
      sj := sj*sjt;
      end
      else
      begin
      sj :=0;
      end;
   end; {loss}

procedure boundary;
   begin
      if structure[position] = 'b' then cnj := cc(cp)
      else cnj := ocp;
      nj := cnj^.r;
      {if G_flag='of' then
      write('b',cnj^.r:5:4,',',-cnj^.i:5:4,' ');}
      if G_flag = 'on' then cnj := ymnj;
      if scan = 'leftt' then
         begin
           neff;
           transmit_admittance;
           loss;
           yj := yjm1;
         end;
   end; {boundary}

procedure thickness;
   begin
      if structure[position] = 't' then lj := rp
      else lj := orp;
      {if G_flag='of' then write('t',lj:5:4,' ');}
      if scan = 'right' then
         begin
           neff;
           transmit_admittance;
           loss;
           yj := yjm1;
         end;
   end; {thickness}

procedure admittance;
   begin
```

```
      if structure[position] = 'a' then aj := cp
      else aj := ocp;
      {if G_flag='of' then write('a',aj^.r:5:3,',',aj^.i:5:3,' ');}
      yjm1 := su(yj,aj);
      sj := sj*(yj^.r/yjm1^.r);
      yj := yjm1;
   end; {admittance}

procedure impedance;
   begin
      if structure[position] = 'z' then zj := cp
      else zj := ocp;
      {if G_flag='of' then write('z',zj^.r:5:4,',',zj^.i:5:4,' ');}
      yjm1 := su(yj,qu(co(1,0),zj));
      sj := sj*(yj^.r/yjm1^.r);
      yj := yjm1;
   end; {admittance}

function FF(t : real) : real;
   begin
      QQ := 1/sqrt(1-sqr(p/l))-1;
      ccc := sqr(cos((pi/2)*(t/p)));
      ss := 1-ccc;
      f1 := QQ*ccc*ccc/(1+QQ*ss*ss);
      f2 := sqr(p*ccc*(1-3*ss)/(4*l));
      FF := f1+f2;
   end;

function LL(tt : real) : real;
   begin
      LL := ln(1/sin((pi*tt)/(2*p)));
   end;

procedure jerusalum_grid;
   begin
      p := rp;
      d := rp;
      v := rp;
      h := rp;
      XX := (p/l)*(LL(v)+FF(v));
      BB := 4*d*(LL(h)+FF(h))/l;
      yjm1 := su(yj,co(0,1/(XX-1/BB)));
      sj := sj*(yj^.r/yjm1^.r);
      yj := yjm1;
   end; {jerusalum_grid}

procedure quasi_static_grid;
   begin
      a := rp;
      g := rp;
      w := rp;
      rs := rp;
      rd := rp;
```

```
      cv := rp;
      bcv := co(0,(2.0*pi*3E11/l)*cv);
      zd := sc(1/377,su(co(rs,0),qu(co(1,0),su(bcv,co(1/rd,0)))));
      {if G_flag='of' then write('g',round(a),' ',round(g),' ',round(w),
                     ' ',round(rs),' ',round(rd),' ',round(cv),' ');}
      bc := co(0,(4*a/l)*((1+nj*nj)/2)*ln(1/sin((pi*g)/(2*a))));
      bl := co(0,-(1/((g/l)*ln(1/(sin((pi*w)/(2*a)))))));
      gj := su(bc,qu(co(1,0),su(zd,qu(co(1,0),bl))));
      xyz := qu(co(1,0),gj);
      yjm1 := su(yj,gj);
      sj := sj*(yj^.r/yjm1^.r);
      yj := yjm1;
   end; {quasi_static_grid}

procedure final_medium;
   begin
      if structure[position] = 'f' then
      begin
        cnj := cc(cp);
        nj := cnj^.r;
      end
      else
      begin
       cnj := cc(ocp);
       nj := cnj^.r;
      end;
      {if G_flag='of' then
      write('f',round(cnj^.r),',',round(-cnj^.i),' ');}
      if G_flag = 'on' then cnj := ymnj;
      neff;
      yj := cnje;
   end; {final_medium}

procedure wavelength;
   begin
      if structure[position] = 'w' then l := rp
      else l := orp;
      {if G_flag='of' then write('w',round(l),' ');}
   end; {wavelength}

procedure te;
   begin
      ao := rp*pi/180;
      p_flag := 'TE';
      pointer := 1;
      cno := cc(cp);
      {if G_flag='of' then write('s',round(ao*180/pi),' ');}
   end; {te}

procedure tm;
   begin
      ao := rp*pi/180;
      p_flag := 'TM';
```

```
      pointer := 1;
      cno := cc(cp);
      {if G_flag='of' then write('p',round(ao*180/pi),' ');}
   end; {tm}

procedure mirror;
   begin
      yj := co(1E3,-1E3);
      {if G_flag='of' then
      write('m',round(yj^.r),',',round(yj^.i),' ');}
   end; {mirror}

procedure parasitic_radiation;
   begin
      par := cp;
      {if G_flag='of' then
      write('r',round(par^.r),',',round(par^.i),' ');}
   end; {parasitic_radiation}

procedure normalization_constant;
   begin
      normalization := rp;
      {if G_flag='of' then write('n',round(normalization),' ');}
   end;

procedure on_off_plot;
   begin
      on_off := trunc(rp);
   end;

procedure eisenhart_kahn_grid;

   label skip;

function sinc2(x, y, z : real) : real;
   begin
      if x = 0 then sinc2 := 1
               else
                 begin
                    sinc := sin(x*pi*y/z)/(x*pi*y/z);
                    sinc2 := sinc*sinc;
                 end;
   end; {sinc square}

procedure com_library;
   begin
     if structure[position] in command_set then begin
        pointer := position;
        case structure[position] of
           'i' : incident_medium;
           'b' : boundary;
           't' : thickness;
           'a' : admittance;
```

```
              'z' : impedance;
              'j' : jerusalum_grid;
              'g' : quasi_static_grid;
              'f' : final_medium;
              'm' : mirror;
              's' : te;
              'p' : tm;
              'w' : wavelength;
              'r' : parasitic_radiation;
              'n' : normalization_constant;
              'o' : on_off_plot;
          end; {case}
       end; {if}
    end; {com_library}

procedure cal_driver;

    begin
    if scan = 'leftt'
       then
          begin
             repeat {decode structure from right to left}
                com_library;
                position := position - 1;
             until position <= positionstop - 1;
          end
       else
          begin
          repeat {decode structure from left to right}
                com_library;
                position := position + 1;
             until position >= positionstop + 1;
          end;
    end; {calculation_driver}

    begin {special_grid}
       a := rp;
       g := rp;
       w := rp;
       rs := rp;
       rd := rp;
       cv := rp;
       bcv := co(0,(2.0*pi*3E11/l)*cv);
       zd := sc(1/377,su(co(rs,0),qu(co(1,0),su(bcv,co(1/rd,0)))));
       {if G_flag='of' then write('G',' ',round(a),' ',round(g),' ',
                                  round(w),' ',round(rs),' ',
                                  round(rd),' ',round(cv),' ');}
       {no choice, must store in real #, freemem cause mess}
       sjsave := sj;
       yjsaver := yj^.r;
       yjsavei := yj^.i;
       cnosaver := cno^.r;
       cnosavei := cno^.i;
```

```
      positionsave := position;
      structuresave := copy(structure,1,150);
      G_flag := 'on';
      positionG := pos('G',structure);
      if pos('f',structure) = 0 then positionf := pos('m',structure)
      else positionf := pos('f',structure);
      bcsum := co(0,0);
      blsum := co(0,0);
      {writeln(lst,sjsave,yjsaver,yjsavei);
      writeln(lst,positionsave,zd^.r,zd^.i);
      writeln(lst,positionG,cnosaver,cnosavei);
      writeln(lst,a,g,w);
      writeln(lst,rs,rd,cv);}
      for m := 0 to 85 do
        begin
          densum := co(0,0);
          for n := 0 to 5 do
            begin
              pointer := 1;
              cnj := cc(cp);
              nj := cnj^.r;
              cnj := ymnj;
              neff;
              cno := cnje;
              scan := 'leftt';
              position := positionf;
              positionstop := positionG;
              cal_driver; write('');
              ymnp := yj;
              position := 1;
              structure[1] := 'f';
              scan := 'right';
              cal_driver; write('');
              ymnm := yj;
              if (m=0) and (n>=1) then begin
                 bcsum := su(bcsum,sc(sinc2(n,g,a),su(ymnp,ymnm)));
                 end;
              if m>=1 then begin
                 en := 2;
                 if n = 0 then en := 1;
                 densum := su(densum,sc(en*sinc2(n,g,a),su(ymnp,ymnm)));
                 end;
            end; {begin of n}
          if m>=1 then begin
              {writeln(lst,'m=',m,'blsum=',blsum^.r,blsum^.i);}
              if m=1 then blsum := co(0,0);
              {writeln(lst,'m=',m,'blsum=',blsum^.r,blsum^.i);}
              blsum := su(blsum,qu(co(sinc2(m,w,a),0),densum));
              {writeln(lst,'m=',m,'blsum=',blsum^.r,blsum^.i);}
              end;
          freemem(b_heap_pointer,-0);
        end; {begin of m}
      yj^.r := yjsaver;
```

```
        yj^.i := yjsavei;
        sj := sjsave;
        cno^.r := cnosaver;
        cno^.i := cnosavei;
        position := positionsave;
        positionstop := 1;
        G_flag := 'of';
        structure := structuresave;
        scan := 'leftt';
        {writeln(lst,sjsave,yj^.r,yj^.i);
        writeln(lst,positionsave,zd^.r,zd^.i);
        writeln(lst,positionG,cno^.r,cno^.i);
        writeln(lst,a,g,w);
        writeln(lst,rs,rd,cv);}
        bcv := co(0,5.65E-4*cv);   {modify the constant if freq change}
        zd := sc(1/377,su(co(rs,0),qu(co(1,0),su(bcv,co(1/rd,0)))));
        {calculating here to avoid freemem over write}
        bc := sc(2*(1-g/a),bcsum);
        bl := qu(co(1,0),sc(2,blsum));
        gj := su(bc,qu(co(1,0),su(zd,qu(co(1,0),bl))));
        xyz := qu(co(1,0),gj);
        yjm1 := su(yj,gj);
        sj := sj*(yj^.r/yjm1^.r);
        yj := yjm1;
    end; {special_grid}

procedure command_library;
    begin
      if structure[position] in command_set then begin
          pointer := position;
          case structure[position] of
             'i' : incident_medium;
             'I' : incident_medium;
             'b' : boundary;
             'B' : boundary;
             't' : thickness;
             'T' : thickness;
             'a' : admittance;
             'A' : admittance;
             'z' : impedance;
             'Z' : impedance;
             'j' : jerusalum_grid;
             'g' : quasi_static_grid;
             'G' : eisenhart_kahn_grid;
             'f' : final_medium;
             'F' : final_medium;
             'm' : mirror;
             's' : te;
             'p' : tm;
             'w' : wavelength;
             'W' : wavelength;
             'd' : dB_plot := true;
             'r' : parasitic_radiation;
```

```
            'n' : normalization_constant;
            'o' : on_off_plot;
         end; {case}
      end; {if}
   end; {command_library}

procedure calculation_driver;

   begin
   if scan = 'leftt'
      then
         begin
            repeat {decode structure from right to left}
               command_library;
               position := position - 1;
            until position <= positionstop - 1;
         end
      else
         begin
         repeat {decode structure from left to right}
               command_library;
               position := position + 1;
            until position >= positionstop + 1;
         end;
   end; {calculation_driver}


{SIMULATION rountines}


procedure simulation;

label quit;
var
next_data : integer;

procedure find_trap;
begin
            b_heap_pointer := heapptr;
            gotoxy(9,23);
            sj := 1;
            yj := co(0,0);
            position := ord(structure[0]);
            positionstop := 1;
            scan := 'leftt';
            G_flag := 'of';
            calculation_driver;
            rc := qu(di(cnoe,yj),su(cnoe,yj));
         reflectance:=ma(pr(su(rc,par),su(cc(rc),cc(par))))/normalization;
            phase := ph(rc);
            transmittance := (1-reflectance)*sj;
            absorptance := 1-transmittance-reflectance;
end;
```

```

procedure accumulate_error;
var
   datavalue : real;
begin
   datavalue := trp[data_index,2];
   ref_err := ref_err + abs( reflectance - datavalue ) ;
end;

{main procedure simulation}
begin
      dB_plot := false;
      write_y_axis;
      structure := edit(14,3,structure);
      x := 1;
      b := 'a';
      par := co(0,0);
      normalization := 1.0;
      optimization_flag := 0;
      on_off := 15;
      ref_err := 0;
      data_index := 1;
      data_inc := round ( int( (xmax-xmin)/(xd*ndata) ) );
      next_data := 1;
      repeat {sweep variable x}
            if (x = next_data) then
               begin
                  if data_index <= ndata then begin
                     find_error_flag := true;
                     find_trap;
                     accumulate_error;
                     find_error_flag := false;
                     data_index := data_index + 1;
                     next_data := next_data + data_inc;
                  end;
               end;
            find_trap;
            if x>=1 then
            begin
            {gotoxy(4,13); write('      ');
            gotoxy(4,13); write(ref_err:6:2);
            gotoxy(4,14); write('      ');
            gotoxy(4,14); write(bc^.i:6:4);
            gotoxy(4,15); write('      ');
            gotoxy(4,15); write(bl^.i:6:4);
            gotoxy(5,16); write('        ');
            gotoxy(5,16); write(zd^.r:8:2);
            gotoxy(5,17); write('        ');
            gotoxy(5,17); write(zd^.i:8:2);
            gotoxy(5,18); write('      ');
            gotoxy(5,18); write(XX:6:4);
            gotoxy(3,19); write('     ');
            gotoxy(3,19); write(sj:3:2);
```

```
            gotoxy(3,20); write('     ');}
            gotoxy(3,20); write(transmittance:3:2);
            gotoxy(3,21); write('     ');
            gotoxy(3,21); write(reflectance:3:2);
            gotoxy(3,22); write('     ');
            gotoxy(3,22); write(absorptance:3:2);
            gotoxy(3,23); write('     ');
            gotoxy(3,23); write(round(phase*180/pi));
            end;
            if x=1 then
            begin
            {gotoxy(1,13); write('er=');
            gotoxy(1,14); write('bc=');
            gotoxy(1,15); write('bl=');
            gotoxy(1,16); write('zdr=');
            gotoxy(1,17); write('zdi=');
            gotoxy(1,18); write('X=');
            gotoxy(1,19); write('s=');}
            gotoxy(1,20); write('t=');
            gotoxy(1,21); write('r=');
            gotoxy(1,22); write('a=');
            gotoxy(1,23); write('p=');
            end;
            i := ((seg(heapptr^) - seg(b_heap_pointer^)) shl 4) +
                 (ofs(heapptr^) - ofs(b_heap_pointer^));
            freemem(b_heap_pointer,i);
            heapptr := ptr(seg(b_heap_pointer^),ofs(b_heap_pointer^));
            freemem(b_heap_pointer,-0);
            {gotoxy(3,4);         what for?}
            delay(10);
            if KeyPressed then read(kbd,b);
            case b of
                'a' : begin
                           plot_result(1);
                           x := x + 1;
                      end;
                'l' : begin
                           plot_result(0);
                           read(kbd,b);
                           x := x - 1;
                      end;
                'r' : begin
                           plot_result(1);
                           read(kbd,b);
                           x := x + 1;
                      end;
               'q' : goto quit;
          end; {case b}
      until x >= ((xmax-xmin) div xd) + 1; {end of x}
      write_x_axis;
      read(kbd,c);
quit:
end;
```

```
{OPTIMIZATION rountines}


function efunc(vertice : glnp) : real;
{find error between predicted and measured}

var
   i : integer;
   error, r, p : real;
   rm : complex;
   q : char;
begin
for i := 1 to ndim do begin
   vert[i] := vertice[i];
end;
error := 0;
optimization_flag := 1;
mark(b_heap_pointer);
for idata := 1 to ndata do begin
nparam := 0;
gotoxy(9,23);
sj := 1.0;
yj := co(0,0);
position := ord(structure[0]);
positionstop := 1;
scan := 'leftt';
G_flag := 'of';
calculation_driver;
rc := qu(di(cnoe,yj),su(cnoe,yj));
r := sqrt(trp[idata,2]);
face := trp[idata,3]*(pi/180.0);
rm := co( r*cos(face) , r*sin(face) );
error := error + ma( di(rc,rm) );
end; {do}
efunc := error;
release(b_heap_pointer);
end;

procedure scalar_range;
begin
   range_variable := cp;
   range[1,range_pointer] := range_variable^.r;
   range[2,range_pointer] := range_variable^.i;
   range_pointer := range_pointer + 1;
end;

procedure complex_range;
begin
   scalar_range;
   if structure[pointer] = ',' then
   begin
```

```
      scalar_range;
   end
   else
   begin
     range[1,range_pointer] := 0;
     range[2,range_pointer] := 0;
     range_pointer := range_pointer + 1;
   end;
end; {of complex_range}


procedure initialize_optimization;

var
   i, j : integer;
   temp : real;

begin

{rightmost parameter goes into col 1, etc. & row 1 takes minimum}
range_pointer := 1;
position := ord(structure[0]);
positionstop := 1;
scan := 'leftt';
repeat
   if structure[position] in command_set then begin
      pointer := position;
      case structure[position] of
         'I' : complex_range;
         'B' : complex_range;
         'T' : scalar_range;
         'A' : complex_range;
         'Z' : complex_range;
         'F' : complex_range;
         'W' : scalar_range;
      end; {of case}
   end; {of if}
   position := position - 1;
until position <= positionstop - 1;
ftol := 0.0001;
iter := 50;
ndim := range_pointer - 1;
mvertices := ndim + 1;

for i := 1 to mvertices do begin
   for j := 1 to ndim do begin
      if i = j then poly[i,j] := range[2,j]
      else poly[i,j] := range[1,j];
   end;
end;

gotoxy(1,6); write('                                             '); gotoxy(1,6);
writeln('intializating ...   ');
```

```
for i := 1 to mvertices do
begin
   for j := 1 to ndim do
   begin
      vert[j] := poly[i,j];
   end;
   y[i] := efunc(vert);
end;
gotoxy(1,7); writeln;
for i := 1 to mvertices do
begin
   for j := 1 to ndim do
   begin
      temp := poly[i,j];
      write(temp:6:4,'  ');
   end;
   temp := y[i];
   write(temp:6:4,'  ');
   writeln;
end;
writeln('iteration cycle #0   ');
writeln;
writeln;
end; {of optimization}

procedure amoeba;

label
   pau, loop;

const
   alpha = 1.0; beta = 0.5; gamma = 2.0; itmax = 500;

var
   mpts, i, j, inhi, ilo, ihi : integer;
   yprr, ypr, rtol : real;
   pr, prr, pbar : glnp;
   q : char;
   temp : real;
begin
   mpts := ndim + 1;
   iter := 0;
loop:
   ilo := 1;
   if ( y[1] > y[2] ) then
   begin
      ihi := 1;
      inhi := 2;
   end
   else begin
      ihi := 2;
      inhi := 1;
   end;
```

```
for i := 1 to mpts do
begin
   if ( y[i] < y[ilo] ) then ilo := i;
   if ( y[i] > y[ihi] ) then begin
      inhi := ihi;
      ihi := i;
   end
   else begin
      if ( y[i] > y[inhi] ) and ( i <> ihi ) then inhi := i;
   end;
end;
rtol := 2.0 * abs( y[ihi]-y[ilo] ) / ( abs(y[ihi]) + abs(y[ilo]) );
if ( rtol < ftol ) then goto pau;
if ( iter = itmax ) then
begin
   writeln('pause in AMOEBA - too many iterations');
   readln(q);
end;
iter := iter + 1;
for j := 1 to ndim do
begin
   pbar[j] := 0.0;
end;
for j := 1 to ndim do
begin
   begin
      for i := 1 to mpts do
      begin
       if ( i <> ihi ) then pbar[j] := pbar[j] + poly[i,j];
      end;
   end;
end;
for j := 1 to ndim do
begin
   pbar[j] := pbar[j]/ndim;
   pr[j] := pbar[j] + alpha * ( pbar[j] - poly[ihi,j] );
end;
ypr := efunc(pr);
if ( ypr <= y[ilo] ) then
begin
    {write('reflection');}
    for j := 1 to ndim do
    begin
       prr[j] := pbar[j] + gamma * ( pr[j] - pbar[j] );
    end;
    yprr := efunc(prr);
    if ( yprr < y[ilo] ) then
    begin
    {writeln(' & expansion');}
       for j := 1 to ndim do
       begin
          poly[ihi,j] := prr[j];
       end;
```

```
          y[ihi] := yprr;
        end
        else
        begin
          {writeln(' only');}
          for j := 1 to ndim do
          begin
            poly[ihi,j] := pr[j];
          end;
          y[ihi] := ypr;
        end;
     end
     else
     begin
       if ( ypr >= y[inhi] ) then
       begin
         if ( ypr < y[ihi] ) then
         begin
         {writeln('line contraction');}
           for j := 1 to ndim do
           begin
             poly[ihi,j] := pr[j];
           end;
           y[ihi] := ypr;
           for j := 1 to ndim do
           begin
             prr[j] := pbar[j] + beta * ( poly[ihi,j] - pbar[j] );
           end;
           yprr := efunc(prr);
           if ( yprr < y[ihi] ) then
           begin
             for j := 1 to ndim do
             begin
               poly[ihi,j] := prr[j];
             end;
             y[ihi] := yprr;
           end;
         end
         else
         begin
           {writeln('surface contraction');}
           for i := 1 to mpts do
           begin
             if ( i <> ilo ) then
             begin
               for j := 1 to ndim do
               begin
                 pr[j] := ( poly[i,j] + poly[ilo,j] ) / 2;
                 poly[i,j] := pr[j];
               end;
             y[i] := efunc(pr);
             end;
           end;
```

```
            end;
         end
         else
         begin
            {writeln('reflection with mild success only');}
            for j := 1 to ndim do
            begin
               poly[ihi,j] := pr[j];
            end;
            y[ihi] := ypr;
         end;
      end;
      writeln;
      for i := 1 to mpts do
      begin
         for j := 1 to ndim do
         begin
            temp := poly[i,j];
            write(temp:6:4,'  ');
         end;
         temp := y[i];
         write(temp:6:4,'  ');
         writeln;
     end;
     writeln('iteration cycle #',iter,'  ');
     writeln;
goto loop;
pau:
end;

procedure optimization;
begin
   dB_plot := false;
   structure := edit(14,3,structure);
   initialize_optimization;
   amoeba;
   writeln; writeln;
   writeln('Optimization finish.');
   writeln('Please record optimized values.');
   writeln('Press Q to Qquit and R to continue.');
end;


{DATA_BASE routines}


procedure graph_units;

begin
   gotoxy(1,2); write('                                                  ');
   write('                                ');
   gotoxy(1,2); write('enter ylmi, ylmx, yrmi, yrmx : ');
   readln(ylmi, ylmx, yrmi, yrmx);
```

```
   write('select curve : (0 = refl), (1 = phase) and (2 = both) ');
   readln(curve_select);
   ylrange := ylmx - ylmi;     ylscale := (ymax-ymin)/ylrange;
   yrrange := yrmx - yrmi;     yrscale := (ymax-ymin)/yrrange;
end;

procedure database;

var
   i : integer;

begin
   gotoxy(31,2); write(' '); read(filename);
   gotoxy(12,3); writeln('inputting data from ',filename);
   assign(data_transfer,filename);
   ndata := 0;
   reset(data_transfer);
   while not eof(data_transfer) do begin
     ndata := ndata + 1;
     readln(data_transfer,trp[ndata,1],trp[ndata,2],trp[ndata,3]);
   end;
   close(data_transfer);
   xmi := trp[1,1];   xmx := trp[ndata,1];
   for i := 2 to ndata do begin
   if trp[i,1] > xmx then xmx := trp[i,1];
   if trp[i,1] < xmi then xmi := trp[i,1];
   end;
   graphwindow (xmin,ymin,xmax,ymax);
   xscale := (xmax - xmin)/(xmx - xmi);
   write_x_coordinates (xmin,ymax+14,(xmax-xmin) div 5,
                        xmi,(xmx-xmi)/5,xmx,4,1);
   write_y_coordinates (xmin-20,ymin+7,(ymax-ymin) div 5,
                        ylmx,(ylmi-ylmx)/5,ylmi,3,1);
   write_y_coordinates (xmax+20,ymin+7,(ymax-ymin) div 4,
                        yrmx,(yrmi-yrmx)/4,yrmi,4,0);
   for i := 1 to ndata do begin
     gotoxy(47,3); writeln(i,'  ',trp[i,1]:4:2,'  ',trp[i,2]:4:2,'  ',
     trp[i,3]:4:2);
     trp1 := trp[i,1] - xmi;
     {plot reflectance}
     if ( curve_select = 0 ) or ( curve_select = 2 ) then begin
        trp2 := ylmx - trp[i,2];
        plot( round(trp1*xscale) - 1 , round(trp2*ylscale) , 1 );
        plot( round(trp1*xscale) , round(trp2*ylscale) , 1 );
        plot( round(trp1*xscale) + 1 , round(trp2*ylscale) , 1 );
        plot( round(trp1*xscale) , round(trp2*ylscale) + 1 , 1 );
        plot( round(trp1*xscale) , round(trp2*ylscale) - 1 , 1 );
     end;
     {plot phase}
     if ( curve_select = 1 ) or (curve_select = 2 ) then begin
        trp3 := yrmx - trp[i,3];
        plot( round(trp1*xscale) - 1 , round(trp3*yrscale) + 1 , 1 );
        plot( round(trp1*xscale) , round(trp3*yrscale ) , 1 );
```

```
            plot( round(trp1*xscale) + 1 , round(trp3*yrscale) - 1 , 1 );
            plot( round(trp1*xscale) + 1 , round(trp3*yrscale) + 1 , 1 );
            plot( round(trp1*xscale) - 1 , round(trp3*yrscale) - 1 , 1 );
         end;
         read(kbd,c);
      end;
      read(kbd,c);
   end;


{MAIN PROGRAM}


begin
ylmi := 0.0;   yrmi := -180.0;
ylmx := 1.0;   yrmx := 180.0;
ylrange := ylmx - ylmi;     ylscale := (ymax-ymin)/ylrange;
yrrange := yrmx - yrmi;     yrscale := (ymax-ymin)/yrrange;
select := 'm';
ndata := 1;
curve_select := 2;
bc := co(0.0,0.0);
bl := co(0.0,0.0);
zd := co(0.0,0.0);
XX := 0.0;
c := 'q';
main :
      repeat
      set_up;
      if select = 'q' then goto pau;
      case select of
         'd' : database;
         's' : simulation;
         'o' : optimization;
         'g' : graph_units;
         'q' : goto pau;
      end;
      if (c = 'q') or (c = 'g') or (c = 'o') then select := 'm';
      if (c = 'd') or (c = 's') then select := c;
      until c = 'q';
      goto main;
pau:
    textmode(bw80);
end.
```

Appendix C

Computer Program Listing for Reflection Measurement

```
program measure_reflection_coefficient;

{       The name of this program is reflmeas. It was developed to
    measure the reflection coefficient of the diode-grid in conjunction
    with the small aperture reflectometer. An IBM-PC is used
    to control the Data Translation A/D and D/A converter and
    the Hp3478A multimeter, whose address should be 23, for
    voltage measurements. When the program is run, it prompts the
    user for vertical plotting units, scanning distance in [mm],
    operating frequency in [GHz], etc. The measured reflection
    coefficients are plotted in real time. At the end of the
    measurement, the data can be saved in a file for later editing
    or processing.     }



type
  complex = ^complex_record;
     complex_record = record
     r,i : real;
     end;
  complex_matrix = ^complex_matrix_record;
     complex_matrix_record = record
     t11,t12,t21,t22 : complex;
     end;
   sdesc = record
           len : byte;
           addr : integer;
           end;
   datatype = array[1..100] of real;

const
   xmin = 190; xmax = 550;
   ymin = 43;  ymax = 163;

label
   pau, restart, next_data;

var
  vout, vin, gain : real;
  channel, dac : integer;
  peak, valley, i1, i2, i3, i4, pin : real;
  amp, face, facex, facey : real;
  ylstart, yrstart, ylstop, yrstop, xstart, xstop : real;
  q, c, select : char;
  dB_plot : boolean;
  on_off : byte;
  x, xcor, ycor : integer;
  norm, min, max : real;
```

```
   reflectance, transmittance, phase, absorptance : real;
   ndata, j : integer;
   lap : array [1..100 , 1..3] of real;
   {1 = len. , 2 = amp. ,and 3 = phase}
   data_transfer : text;
   filename : string[15];
   port_num, bit_num, bit_val, port_0_byte, port_1_byte : byte;
   n_step, over_drive_n_step : integer;
   n_step_per_um, freq, inc_90, lamda : real;
   lmirror, lprevious, lphase, lstart, lstep, lstop : real;
   i, mode, p_step, m_step, total_step  : integer;
   cmd, recv, set_up_store : string[200];
   cmddesc, recvdesc : sdesc;
   my_address, system_controller, len, status, code : integer;
   quote : char;
   offset, pnorm, initial_sep : real;


function co (s,t : real) : complex; {makes a complex number}
var
   u : complex;
begin
   new(u);
   u^.r := s;
   u^.i := t;
   co := u;
end;

function ph(s : complex) : real; {phase in radians, (-pi,+pi]}
begin
   if s^.r > 0 then ph := arctan(s^.i/s^.r);
   if s^.r < 0 then if s^.i >= 0 then ph := arctan(s^.i/s^.r) + pi
      else ph := arctan(s^.i/s^.r) - pi;
   if s^.r = 0 then begin
         if s^.i > 0 then ph := pi/2;
         if s^.i < 0 then ph := -pi/2;
         if s^.i = 0 then ph := 0;
      end; {real part 0}
end; {of phase}

procedure draw_box (x1,y1,x2,y2,color : integer);
begin
   draw (x1,y1,x1,y2,1);
   draw (x1,y2,x2,y2,1);
   draw (x2,y2,x2,y1,1);
   draw (x2,y1,x1,y1,1);
end;

procedure draw_x_ticks(x1,y1,x_increment,x2 : integer);
begin
   while (x1<=x2) do begin
   draw (x1,(y1+1),x1,(y1-2),1);
   x1 := x1 + x_increment;
```

```
   end; {while}
end; {draw_x_ticks}

procedure draw_y_ticks(x1,y1,y_increment,y2 : integer);
begin
   while (y1<=y2) do begin
   draw ((x1+2),y1,(x1-4),y1,1);
   y1 := y1 + y_increment;
   end; {while}
end; {draw_y_ticks}

procedure draw_graph(x1,y1,x2,y2,
   x_increment, y_increment_left, y_increment_right :integer);
begin
   draw_box(x1,y1-2,x2,y2+2,1);
   draw_x_ticks (x1,y1,x_increment,x2);
   draw_x_ticks (x1,y2,x_increment,x2);
   draw_y_ticks (x1,y1,y_increment_left,y2);
   draw_y_ticks (x2,y1,y_increment_right,y2);
end; {procedure draw_graph}

procedure write_x_coordinates(x1,y1,x_increment :integer;
   x1c,xc_increment,x2c : real; field,fix :integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (x1c:field:fix);
   x1c := x1c + xc_increment;
   x1 := x1 + x_increment;
   until x1c > x2c;
end; {write_x_coordinates}

procedure write_y_coordinates(x1,y1,y_increment :integer;
   y1c,yc_increment,y2c : real; field, fix : integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (y1c:field:fix);
   y1c := y1c + yc_increment;
   y1 := y1 + y_increment;
   until y1c < y2c;
end; {write_y_coordinates}

function dB(x : real) : real; {give the dB difference}
begin
dB := 10*ln(x)/ln(10);
end; {dB}

procedure set_up;
begin
       hires; hirescolor(15);
       draw_graph(xmin,ymin,xmax,ymax,(xmax-xmin)div 5,
                  (ymax-ymin)div 5, (ymax-ymin)div 4);
```

```
      gotoxy(1,7); write ('Pha.   xxxxx');
      gotoxy(1,8); write('Refl.  +++++ ');
end; {set_up}

procedure write_y_axis;
begin
   write_y_coordinates(xmax+20,ymin+7,(ymax-ymin) div 4,
                       yrstop,(yrstart-yrstop)/4,yrstart,4,0);
   if dB_plot then
      write_y_coordinates(xmin-20,ymin+7,(ymax-ymin) div 5,
                          0,-10,-50,3,0)
      else
      write_y_coordinates(xmin-40,ymin+7,(ymax-ymin) div 5,
                          ylstop,(ylstart-ylstop)/5,ylstart,3,1);
end; {write_y_axis}

procedure write_x_axis;
begin
   write_x_coordinates (xmin,ymax+14,(xmax-xmin) div 5,
                        xstart,(xstop-xstart)/5,xstop,4,1);
end; {write_x_axis}

procedure initialize(var addr, level : integer);
   external 't488init';

procedure transmit(var s : sdesc; var status : integer);
   external 't488xmit';

procedure receive(var r : sdesc; var len, status : integer);
   external 't488recv';

procedure hp_vm_initialize;
{Initalization procedure for Hp voltmeter}
begin
   quote := chr(39);
   cmddesc.addr := ofs(cmd) + 1;
   recvdesc.addr := ofs(recv) + 1;

   my_address := 21;
   system_controller := 0;
   initialize(my_address, system_controller);

   cmd := 'LISTEN 23' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA ' + quote + 'H1 T1' + quote + '13 10' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'TALK 23' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);
```

```
end;

procedure hp_vm(var voltage : real);

begin
      recv := '                                                            ';
      recvdesc.len := length(recv);
      receive(recvdesc,len,status);

      recv := '                                                            ';
      recvdesc.len := length(recv);
      receive(recvdesc,len,status);
      if recv[1] = '+' then delete(recv,1,1);
      val(recv,voltage,code);
      voltage := ( voltage - offset )* 1e3 ;

end; {of hp_vm}

procedure wait_for_DIF;
{check data register of Data Translation board before writing to it}

var
     ready,int : byte;
     timeout : integer;

begin
     ready := 0;
     timeout := $8000;
     while ready = 0 do
          begin
               int := Port[$2ED] and 2;
               if int = 0 then ready := 1;
               delay(1);
               timeout := timeout+1;
               if timeout = 32767 then
                    begin
                    writeln('Device time-out on DT-2801 board,');
                    writeln('during wait for DIF.');
                    writeln;
                    ready := 1;
               end;{if}
          end;{while}
end;{procedure wait_for_DIF}


procedure wait_for_DOR;
{check if there is new data in Data Translation board}

var
     ready,int : byte;
     timeout : integer;

begin
```

```
     ready := 0;
     timeout := $8000;
     while ready = 0 do
          begin
               int := Port[$2ED] and 1;
               if int = 1 then ready := 1;
               delay(1);
               timeout := timeout+1;
               if timeout = 32767 then
                    begin
                    writeln('Device time-out on DT-2801 board,');
                    writeln('during wait for DOR.');
                    writeln;
                    ready := 1;
               end;{if}
          end;{while}
end;{procedure wait_for_DOR}


procedure wait_for_ready;
{check if ready to execute another command in the
Data Tranlation board}

var
     ready,int : byte;
     timeout : integer;

begin
     ready := 0;
     timeout := $8000;
     while ready = 0 do
          begin
               int := Port[$2ED] and 4;
               if int = 4 then ready := 1;
               delay(1);
               timeout := timeout+1;
               if timeout = 32767 then
                    begin
                    writeln('Device time-out on DT-2801 board,');
                    writeln('during wait for Ready.');
                    writeln;
                    ready := 1;
               end;{if}
          end;{while}
end;{procedure wait_for_ready}


procedure write_d_to_a_immediate(vout: real; dac: integer);

var
     dat_low_byte, dat_high_byte, ps : byte;
     code : integer;

```

```
begin
     wait_for_ready;
     Port[$2ED] := $08;  {write voltage command}
     code := round((vout+10)*4095/20.0);
     if code > 4095 then code := 4095;
     if code < 0 then code := 0;
     dat_high_byte := code and $0F00 shr 8;
     dat_low_byte := code and $FF;
     if dac = 0 then ps := $00;
     if dac = 1 then ps := $01;
     wait_for_DIF;
     wait_for_DIF;
     Port[$2EC] := ps;
     wait_for_DIF;
     Port[$2EC] := dat_low_byte;
     wait_for_DIF;
     Port[$2EC] := dat_high_byte;
end;{procedure write_d_to_a_immediate}


procedure read_a_to_d_immediate(var vin, gain : real; channel : integer);
{Read data form the Data Translation board}
var
     dat_low_byte, dat_high_byte, G, chan : byte;
     code : integer;
     temp1 : real;
     i : integer;

begin
     temp1 := 0;
     {for i := 1 to 10 do begin}
     wait_for_ready;
     port[$2ED] := $0C;  {A/D command}
     wait_for_DIF;
     if gain = 1.0    then G := $00;
     if gain = 10.0   then G := $01;
     if gain = 100.0  then G := $02;
     if gain = 500.0  then G := $03;
     port[$2EC] := G;  {Gain select}
     wait_for_DIF;
     case channel of
     0 : chan := $00;
     1 : chan := $01;
     2 : chan := $02;
     3 : chan := $03;
     4 : chan := $04;
     5 : chan := $05;
     6 : chan := $06;
     7 : chan := $07;
     end;
     port[$2EC] := chan;  {Channel select}
     wait_for_DOR;
     dat_low_byte := port[$2EC];
```

```
     wait_for_DOR;
     dat_high_byte := port[$2EC];
     code := 256 * dat_high_byte + dat_low_byte;
     temp1 := temp1 + 20.0 * code/65536.0;
     vin := temp1/gain;
end;{procedure read_a_to_d_immediate}


procedure clear_DT_2801;
{Clear procedure for Data Translation board}
var
     temp : byte;

begin
     Port[$2ED] := $FF;
     temp := Port[$2EC];
     wait_for_ready;
     Port[$2ED] := 0;
     wait_for_DOR;
     temp := Port[$2EC];
end;{procedure clear_DT_2801}

procedure wait_for_keypress;

begin
   if mode = 0 then read(kbd,q)
   else delay(2000);
end;{procedure wait_for_keypress}

procedure write_digital_out(port_number, bit_number, on_off : byte);
{Write a command to motor driver}

var
     mask : byte;
begin
     wait_for_ready;
     port[$2ED] := $05;
     wait_for_DIF;
     port[$2EC] := port_number;
     wait_for_ready;
     port[$2ED] := $07;
     wait_for_DIF;
     port[$2EC] := port_number;
     wait_for_DIF;
     mask := 1 shl bit_number;
     if port_number = 1 then
        begin
           if on_off = 1 then port_1_byte := mask or port_1_byte
           else port_1_byte := (not mask) and port_1_byte;
           port[$2EC] := port_1_byte;
        end
     else
        begin
```

```
            if on_off =1 then port_0_byte := mask or port_0_byte
            else port_0_byte := (not mask) and port_0_byte;
            port[$2EC] := port_0_byte;
         end;
end;

procedure enable_motor(port_number, on_off : byte);
{Enable motor: 1 for on }

begin
    if on_off = 1 then write_digital_out(port_number, 2, 1)
    else write_digital_out(port_number, 2, 0);
end;

procedure direction(port_number, for_bak : byte);
{Motor direction: 1 for forward }

begin
   if for_bak = 1 then write_digital_out(port_number, 5, 1)
   else write_digital_out(port_number, 5, 0);
end;

procedure scan(port_number : byte ; n_step : integer);
{Move motor: 1 for move}
var
   i : integer;

begin
   if n_step > 0 then direction(port_number,1)
   else direction(port_number,0);
   for i := 1 to abs(n_step) do begin
   write_digital_out(port_number, 7, 0);
   write_digital_out(port_number, 7, 1);
   delay (20);
   write_digital_out(port_number, 7, 0);
   end;
end; {of scan}

begin {main program}
   hp_vm_initialize;
   dB_plot := false;
   set_up;
   gotoxy(1,1); write('enter ylstart, ylstop, yrstart and yrstop : ');
   readln(ylstart,ylstop,yrstart,yrstop);
   write_y_axis;
   gotoxy(1,2); write('enter xstart and xstop : ');
   readln(xstart,xstop);
   gotoxy(1,3); write('enter mode ( manual = 0  &  auto = 1 ) : ');
   readln(mode);
   write_x_axis;

   port_0_byte := 0;
   port_1_byte := 0;
```

```
   clear_DT_2801;
   enable_motor(0,1);
   enable_motor(1,1);
   n_step_per_um := 1.01;    {Motor calibration constant}
   {Last calibrated on 10/86}
   offset := 0.10e-3;
   over_drive_n_step := 25;
   ndata := 1;
   total_step := 0;
   q := 'z';

   gotoxy(1,3); writeln('                                                  ');
   gotoxy(1,3); write('enter frequency in GHz : ');
   readln(freq);
   lamda := 300.0/freq;   {convert into millimeter}
   inc_90 := lamda/8.0;
   p_step := round( inc_90 * 1000.0 * n_step_per_um );

   gotoxy(60,3); write('STATE :');
   if mode = 1 then begin
      gotoxy(1,4); write('enter lstart, lstep, lstop : ');
      readln(lstart, lstep, lstop);
      lmirror := lstart;
      lprevious := lmirror;
   end
   else begin
      gotoxy(70,3); write('L  ');
      gotoxy(1,14); write('  L >          ');
      gotoxy(7,14); readln(lmirror);
      lprevious := xstart;
   end;
   hp_vm_initialize;
   {To eliminate back-lash in the translation}
   scan(0,-over_drive_n_step);
   scan(0,over_drive_n_step);
   scan(0,-over_drive_n_step);
   scan(0,over_drive_n_step);
   scan(1,-over_drive_n_step);
   scan(1,over_drive_n_step);
   scan(1,-over_drive_n_step);
   scan(1,over_drive_n_step);
   gotoxy(1,4);
   writeln('                                                  ');
   gotoxy(1,4); write('enter initial seperation : ');
   readln(initial_sep);
writeln(lst,'initial seperation = ',initial_sep);
writeln(lst,' ');
write(lst,'   L         Pin        i1       ');
writeln(lst,'i2         i3        i4       Phase     Ampc');
writeln(lst,' ');
next_data:
   restart:
      if ( mode = 0 ) and ( ndata <> 1 ) then begin
```

```
          gotoxy(70,3); write('L  ');
          gotoxy(1,14); write('  L >          ');
          gotoxy(7,14); readln(lmirror);
       end;
       m_step := round( (lmirror-lprevious) * 1000.0 * n_step_per_um );
       scan(0,m_step);
       gotoxy(70,3); write('L  ');
       gotoxy(1,14); write('  L = ');
       if mode = 1 then write(lmirror:6:3);

       gain := 1.0;
       channel := 0;
       gotoxy(70,3); write('Pin');
       gotoxy(1,15); write('Pin =          ');
       wait_for_keypress;
       {hp_vm(pin);}
       read_a_to_d_immediate(pin, gain, channel);
       gotoxy(7,15); write(pin:7:6);
       gotoxy(70,3); write('i1 ');

       gotoxy(1,16); write(' i1 =          ');
       wait_for_keypress;
       hp_vm(i1);
       read_a_to_d_immediate(pnorm, gain, channel);
       i1 := i1/pnorm;
       gotoxy(7,16); write(i1:7:6);

       gotoxy(70,3); write('i2 ');
       scan(1,p_step);
       gotoxy(1,17); write(' i2 =          ');
       wait_for_keypress;
       hp_vm(i2);
       read_a_to_d_immediate(pnorm, gain, channel);
       i2 := i2/pnorm;
       gotoxy(7,17); write(i2:7:6);

       gotoxy(70,3); write('i3 ');
       scan(1,p_step);
       gotoxy(1,18); write(' i3 =          ');
       wait_for_keypress;
       hp_vm(i3);
       read_a_to_d_immediate(pnorm, gain, channel);
       i3 := i3/pnorm;
       gotoxy(7,18); write(i3:7:6);

       gotoxy(70,3); write('i4 ');
       scan(1,p_step);
       gotoxy(1,19); write(' i4 =          ');
       wait_for_keypress;
    if q='r' then goto restart;
       hp_vm(i4);
       read_a_to_d_immediate(pnorm, gain, channel);
       i4 := i4/pnorm;
```

```
      gotoxy(7,19); write(i4:7:6);

      facey := i4-i2;
      facex := i1-i3;
      face := -ph(co(facex,facey))*(180/pi);
      gotoxy(1,20); write('Pha =          ');
      gotoxy(7,20); write(face:4:1);
      xcor := round((lmirror-xstart)*(xmax-xmin)/(xstop-xstart));
      ycor := round((yrstop-face)*(ymax-ymin)/(yrstop-yrstart));
      graphwindow (xmin,ymin,xmax,ymax);
      plot(xcor-1,ycor+1,1);
      plot(xcor+1,ycor+1,1);
      plot(xcor,ycor,1);
      plot(xcor+1,ycor-1,1);
      plot(xcor-1,ycor-1,1);

      amp := sqrt(sqr(i4-i2)+sqr(i1-i3));
      gotoxy(1,21); write('Amp =         ');
      gotoxy(7,21); write(amp:5:3);
      xcor := round((lmirror-xstart)*(xmax-xmin)/(xstop-xstart));
      ycor := round((ylstop-amp)*(ymax-ymin)/(ylstop-ylstart));
      graphwindow(xmin,ymin,xmax,ymax);
      plot(xcor-1,ycor,1);
      plot(xcor+1,ycor,1);
      plot(xcor,ycor-1,1);
      plot(xcor,ycor+1,1);

      write(Lst,lmirror:6:4,'   ');
      write(Lst,pin:7:6,'   ');
      write(Lst,i1:7:6,'   ');
      write(Lst,i2:7:6,'   ');
      write(Lst,i3:7:6,'   ');
      write(Lst,i4:7:6,'   ');
      write(Lst,face:4:1,'   ');
      write(Lst,amp:6:3);
      writeln(Lst,' ');

      lap[ndata,1] := lmirror;
      lap[ndata,2] := amp;
      lap[ndata,3] := face;
      total_step := total_step + m_step;
      if ( mode = 1 ) and ( (lmirror>lstop) or (lmirror=lstop) )
      then q := 's';
      if q='s' then begin
           gotoxy(1,24); write('enter impedance filename : ');
           readln(filename);
           assign(data_transfer,filename);
           rewrite(data_transfer);
           for j := 1 to ndata do
           begin
              writeln(data_transfer,lap[j,1],' ',lap[j,2],' ',lap[j,3]);
           end;
           close(data_transfer);
```

```
            writeln(lst,' ');
            writeln(lst,'stored in filename : ',filename);
            goto pau;
        end; {of begin}

        if q = 'q' then goto pau;
        gotoxy(70,3); write('Rwd');
        scan(1,-3*p_step);
        scan(1,-over_drive_n_step);
        scan(1,over_drive_n_step);
        lprevious := lmirror;
        if mode = 1 then lmirror := lmirror + lstep;
        ndata := ndata + 1;
    goto next_data;
pau:
        {Return to original positon}
        scan(0,-total_step);
        scan(0,-over_drive_n_step);
        scan(0,over_drive_n_step);
        scan(1,-3*p_step);
        scan(1,-over_drive_n_step);
        scan(1,over_drive_n_step);
        enable_motor(0,0);
        enable_motor(1,0);
        textmode(bw80);
        gotoxy(1,1);
end.
```

Appendix D

Computer Program Listing for Diode Parameter Measurement

```
program iv_measurement_of_Schottky_diode;


{       The name of this program is ivdiode. It was developed for
     measuring the circuit parameters of a Schottky diode.
     The program is written in Turbo Pascal. An IBM-PC is used
     to control the Hp4145A semiconductor parameter analyzer,
     whose address is 19. The diode leads should be connected
     to the terminals labeled as SMU1, and GND on the I-V box
     are used. The diode parameters are calculated from the measured
     currents and voltages. They are displayed on the monitor.
     The diode parameters include the series resistance (Rs),
     barrier height (Vb), n-factor (nf), and saturation current
     (Io). The correlation coefficient (cc) of the fit is also
     displayed. The algorithm used in this calculation
     is given in Chung-en Zah's Ph.D. thesis, "Millimeter-Wave
     Monolithic Schottky Diode Imaging Arrays." The program is
     menu driven. Default value for the diode area is 18 um*2.
     The current ranges from 0.1 uA to 0.1 mA, and the voltage
     ranges from 0 V to 2 V. Pressing return after the prompt
     invokes the above default values. The data can be saved on disk,
     retrived from disk, displayed on the monitor, or listed on the
     printer.        }

type
   sdesc = record
           len : byte;
           addr : integer;
           end;
   datarray = array [1..100] of real;

const
   xmin = 190; xmax = 450;
   ymin = 23;  ymax = 163;
   xd = 3;

label
   quit, main_menu, sub_menu, loop1, loop2;

var
   cmd, recv : string[200];
   cmddesc, recvdesc : sdesc;
   my_address, system_controller, len, status : integer;
   quote : char;
   first, nchar, code: integer;
   voltage, current, va, im, vm : real;
   filename, buffer : string[10];
   main_mode, sub_mode, q : char;
```

```
   time_delay, j, ndata : integer;
   v, i : datarray;
   ia, ylstart, ylstop, xstart, xstop, ydata : real;
   xcor, ycor, yb, ye : integer;
   vb, area, rs, nf, gf, is : real;
   number_transfer : file of real;
   row, col : real;

procedure initialize(var addr, level : integer);
   external 't488init';

procedure transmit(var s : sdesc; var status : integer);
   external 't488xmit';

procedure receive(var r : sdesc; var len, status : integer);
   external 't488recv';

procedure initialize_hp_4145a;

begin
   time_delay := 4000;
   quote := chr(39);
   cmddesc.addr := ofs(cmd) + 1;
   recvdesc.addr := ofs(recv) + 1;

   my_address := 21;
   system_controller := 0;
   initialize(my_address, system_controller);

   cmd := 'IFC REN MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'US' + quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'IT2 CA1 BC' + quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);
end;

procedure reset_hp_4145a;

begin
   cmd := 'MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'DCL SDC' + quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);
end; {of reset_hp_4145a}
```

```
procedure set_voltage(volt : real);

var
  v : string[18];

begin
   str(volt,v);  delete(v,8,6);
   cmd := 'MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'DV 1, 0, ' + v + ', 10E-3' +
   quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   delay(time_delay);
end; {of set_voltage}

procedure set_current(ampere : real);

var
  i : string[18];

begin
   str(ampere,i);  delete(i,8,6);
   cmd := 'MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'DI 1, 0, ' + i + ', 5' +
   quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   delay(time_delay);
end; {of set_voltage}

procedure measure_current(var i : real);

begin
   cmd := 'MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'TI 1' + quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'MLA TALK 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);
```

```

   recv := '                              ';
   recvdesc.len := length(recv);
   receive(recvdesc,len,status);
   delete(recv,1,3);
   val(recv,current,code);
   if current < 0 then i := current
   else begin
     delete(recv,1,1);
     val(recv,current,code);
     i := current;
   end;
end; {of measure_current}

procedure measure_voltage(var v : real);

begin
   cmd := 'MTA LISTEN 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA' + quote + 'TV 1' + quote + '13 10';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'MLA TALK 19';
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   recv := '                              ';
   recvdesc.len := length(recv);
   receive(recvdesc,len,status);
   delete(recv,1,3);
   val(recv,voltage,code);
   if voltage < 0 then v:= voltage
   else begin
      delete(recv,1,1);
      val(recv,voltage,code);
      v := voltage;
   end;
end; {measure_voltage}

procedure clear_line;
begin
gotoxy(1,1); write('                                        ');
             write('                                        ');
end; {clear_line}

procedure display(x, y : datarray; ndata : integer);
var
   cont : char;
   i : integer;
begin
```

```
   clear_line;
   for i := 1 to ndata do begin
   gotoxy(1,1);   write(ndata,' - ',i,'  ',y[i]:10,'  ',x[i]:10);
   read(cont);
   end;
end; {of display}

procedure lsf(x, y : datarray; n : integer; var a, b, c  : real);
{a = slope     b = intercept     c = correlation coeff. }
var
   xsum, ysum, xysum, xssum, yssum : real;
   xden, yden, rden : real;
   i : integer;

begin
   xsum := 0; ysum := 0;
   xysum := 0;
   xssum := 0; yssum := 0;
   for i := 1 to n do begin
      xsum := xsum + x[i];
      ysum := ysum + y[i];
      xysum := xysum + x[i]*y[i];
      xssum := xssum + x[i]*x[i];
      yssum := yssum + y[i]*y[i];
   end;
   xden := n * xssum - xsum * xsum;
   yden := n * yssum - ysum * ysum;
   rden := sqrt( xden * yden );
   a := ( n*xysum - xsum*ysum ) / xden;
   b := ( ysum*xssum - xysum*xsum )/ xden;
   c := ( n*xysum - xsum*ysum ) / rden;
end; {of lsf}

procedure curve_fit_ivdata(x, y : datarray; n : integer;
                       var rs, nf, gf, is : real);
{ x = voltage   y = current    gf = goodness }
var
   lx, ly, yav, dvdlni : datarray;
   i : integer;
begin
   {for i := 1 to n-1 do begin
      dvdlni[i] := ( x[i+1]-x[i] ) / ( ln(y[i+1])-ln(y[i]) );
      yav[i] := ( y[i+1] + y[i] ) / 2.0 ;
   end;
   lsf(yav, dvdlni, n-1, rs, nf, gf);
   nf := nf/0.0259; This method is not accurrate because yav is used.}
   for i := 1 to n-1 do begin
      lx[i] := ( y[i+1]-y[i] ) / ( x[i+1]-x[i] );
      ly[i] := ( ln(y[i+1])-ln(y[i]) ) / ( x[i+1]-x[i] );
   end;
   lsf(lx, ly, n-1, rs, nf, gf);
   rs := -rs/nf;
   nf := 1/(0.0259*nf);
```

```
   is := 0.0;
   for i := 1 to n do begin
      is := is + ln(y[i]) - (x[i]-y[i]*rs)/(nf*0.0259);
   end;
   is := exp(is/n);
   gotoxy(1,7); write('Rs = ',rs:8);
   gotoxy(1,8); write('nf = ',nf:8);
   gotoxy(1,9); write('Io = ',is:8);
   gotoxy(1,10); write('cc = ',gf:8);
end; {curve_fit_ivdata}


procedure draw_box (x1,y1,x2,y2,color : integer);
begin
   draw (x1,y1,x1,y2,1);
   draw (x1,y2,x2,y2,1);
   draw (x2,y2,x2,y1,1);
   draw (x2,y1,x1,y1,1);
end; {of draw_box}

procedure draw_x_ticks(x1,y1,x_increment,x2 : integer);
begin
   while (x1<=x2) do begin
   draw (x1,(y1+1),x1,(y1-2),1);
   x1 := x1 + x_increment;
   end; {while}
end; {of draw_x_ticks}

procedure draw_y_ticks(x1,y1,y_increment,y2 : integer);
begin
   while (y1<=y2) do begin
   draw ((x1+2),y1,(x1-4),y1,1);
   y1 := y1 + y_increment;
   end; {while}
end; {of draw_y_ticks}

procedure draw_graph(x1,y1,x2,y2,
   x_increment, y_increment_left, y_increment_right :integer);
begin
   draw_box(x1,y1-2,x2,y2+2,1);
   draw_x_ticks (x1,y1,x_increment,x2);
   draw_x_ticks (x1,y2,x_increment,x2);
   draw_y_ticks (x1,y1,y_increment_left,y2);
   draw_y_ticks (x2,y1,y_increment_right,y2);
end; {of procedure draw_graph}

function ten_to_power(n : integer) : real;
begin
   ten_to_power := exp( n*ln(10) );
end; {of ten_to_power}

function ten_to(n : real) : real;
begin
```

```
   ten_to := exp( n*ln(10) );
end; {of ten_to}

procedure write_x_coordinates(x1,y1,x_increment :integer;
   x1c,xc_increment,x2c : real; field,fix :integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (x1c:field:fix);
   x1c := x1c + xc_increment;
   x1 := x1 + x_increment;
   until x1c > x2c;
end; {of write_x_coordinates}

procedure write_y_coordinates(x1,y1,y_increment :integer;
   y1c,yc_increment,y2c : real; field, fix : integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (y1c:field:fix);
   y1c := y1c + yc_increment;
   y1 := y1 + y_increment;
   until y1c < y2c;
end; {of write_y_coordinates}

procedure set_up;
begin
      hires; hirescolor(15);
      draw_graph(xmin,ymin,xmax,ymax,(xmax-xmin)div 5,
                 (ymax-ymin)div 6, (ymax-ymin)div 6);
end; {of set_up}

procedure write_y_axis;
begin
   write_y_coordinates(xmin-40,ymin+7,(ymax-ymin) div 6,
                       ylstop,(ylstart-ylstop)/6,ylstart,4,0);
end; {of write_y_axis}

procedure write_x_axis;
begin
   write_x_coordinates (xmin,ymax+14,(xmax-xmin) div 5,
                        xstart,(xstop-xstart)/5,xstop,4,1);
end; {of write_x_axis}

procedure plot_ivdata(x, y : datarray; n : integer);
var
   i : integer;
begin
   display(x, y, n);
   set_up;
   gotoxy(1,1); write('enter ylstart ylstop xstart xstop : ');
   readln(ylstart, ylstop, xstart, xstop);
   write_y_axis; write_x_axis;
```

```
    for i := 1 to n do begin
       xcor := round((x[i]-xstart)*(xmax-xmin)/(xstop-xstart));
       ycor := round((ylstop-ln(y[i])/ln(10.0))*
               (ymax-ymin)/(ylstop-ylstart));
       graphwindow(xmin,ymin,xmax,ymax);
       plot(xcor-1,ycor,1);
       plot(xcor+1,ycor,1);
       plot(xcor,ycor-1,1);
       plot(xcor,ycor+1,1);
    end;
end; {plot_datafile}

procedure plot_ivcurve;
var
  i, yd : integer;
  amp, l, xc : real;
  cont : char;
begin
      xc := 1.0;
      yd :=1;
      repeat
         amp := ylstart + (ylstop-ylstart)*xc*yd/(ymax-ymin);
         amp := ten_to(amp);
         l := 0.0259*nf*ln(amp/is + 1) + amp*rs;
         {clear_line;
         gotoxy(1,1); write(l:8,'  ',amp:8);
         read(cont);}
         amp := ln(amp)/ln(10.0);
         xcor := round((l-xstart)*(xmax-xmin)/(xstop-xstart));
         ycor := round((ylstop-amp)*(ymax-ymin)/(ylstop-ylstart));
         graphwindow(xmin,ymin,xmax,ymax);
         plot(xcor,ycor,1);
         xc := xc + 1.0;
      until xc >= int(((ymax-ymin) div yd) + 1);
end; {of plot_datafile}

procedure readfile(var x, y : datarray; var ndata : integer);
var
   i : integer;
   data_transfer : text;
begin
   clear_line;
   gotoxy(1,1); write('enter filename(read) : ');
   readln(filename);
   assign(data_transfer,filename);
   ndata := 0;
   reset(data_transfer);
   while not eof(data_transfer) do begin
     ndata := ndata + 1;
     readln(data_transfer,x[ndata],y[ndata]);
   end;
   close(data_transfer);
end; {of readfile}
```

```
procedure writefile(x, y : datarray; ndata : integer);
var
   i : integer;
   data_transfer : text;
begin
   clear_line;
   gotoxy(1,1); write('enter filename(write) : ');
   readln(filename);
   assign(data_transfer,filename);
   rewrite(data_transfer);
   for i := 1 to ndata do
   begin
      writeln(data_transfer,x[i],'  ',y[i]);
   end;
   close(data_transfer);
end; {of writefile}

procedure convert_lin_to_log(var z : datarray;
                             ndata : integer; q : char);
var
   i : integer;
begin
   clear_line;
   gotoxy(1,1); write('taking log(',q,') ... ');
   for i := 1 to ndata do begin
      z[i] := ln(z[i])/ln(10.0);
   end;
   write('done');
   read(q);
end; {of convert_lin_to_log}

procedure calculate(var x, y : datarray; ndata : integer);
var
   i : integer;
begin
   clear_line;
   gotoxy(1,1); write('log(x) or log(y) : ');
   read(q);
   if q = 'x' then convert_lin_to_log(x, ndata, q);
   if q = 'y' then convert_lin_to_log(y, ndata, q);
end; {of calculate}

procedure list(x, y : datarray; ndata : integer);
var
   i : integer;
begin
   writeln(lst,'filename : ',filename);
   for i := 1 to ndata do begin
      writeln(lst,i,'  ',y[i]:10,'  ',x[i]:10);
   end;
end; {of writedata}
```

```
procedure plot_a_point(x, y : real);
begin
   xcor := round((x-xstart)*(xmax-xmin)/(xstop-xstart));
   ycor := round((ylstop-y)*(ymax-ymin)/(ylstop-ylstart));
   graphwindow(xmin,ymin,xmax,ymax);
   plot(xcor-1,ycor,1);
   plot(xcor+1,ycor,1);
   plot(xcor,ycor-1,1);
   plot(xcor,ycor+1,1);
end; {plot_a_point}

procedure test(var x, y : datarray; ndata : integer);
var
   i_zero : real;
begin
   ndata := 0;
   i_zero := 0.0;
   set_up;
   gotoxy(1,1); write('enter ylstart ylstop xstart xstop : ');
   readln(ylstart, ylstop, xstart, xstop);
   write_y_axis;
   write_x_axis;
   initialize_hp_4145a;
   set_current(i_zero);
   for j := trunc(ylstart)+1 to trunc(ylstop)-1 do begin
      ndata := ndata + 1;
      ia := 1.0 * ten_to_power(j);
      set_current(ia);
      measure_current(y[ndata]);
      measure_voltage(x[ndata]);
      ydata := ln(y[ndata])/ln(10.0);
      plot_a_point(x[ndata],ydata);

      ndata := ndata + 1;
      ia := 2.0 * ten_to_power(j);
      set_current(ia);
      measure_current(y[ndata]);
      measure_voltage(x[ndata]);
      ydata := ln(y[ndata])/ln(10.0);
      plot_a_point(x[ndata],ydata);

      ndata := ndata + 1;
      ia := 4.0 * ten_to_power(j);
      set_current(ia);
      measure_current(y[ndata]);
      measure_voltage(x[ndata]);
      ydata := ln(y[ndata])/ln(10.0);
      plot_a_point(x[ndata],ydata);
   end;
   set_current(i_zero);
   curve_fit_ivdata(x, y, ndata, rs, nf, gf, is);
end; {of test}
```

```
procedure find_barrier_height;
begin
   clear_line;
   gotoxy(1,1); write('Enter diode area : '); readln(area);
   vb := (-0.0259) * ln( is / (area*8.16*sqr(300)) );
   gotoxy(1,12); write('Vb = ',vb:8);
end;

procedure create_new_buffer;
var
   select : integer;

begin
   clear_line;
   gotoxy(1,1);
   write('Enter 1 for create & 0 for append to a file : ');
   read(select);
   clear_line;
   gotoxy(1,1); write('Enter filename : '); readln(buffer);
   if select = 1 then
   begin
   assign(number_transfer,buffer);
   rewrite(number_transfer);
   close(number_transfer);
   end;
   write(lst,'Row  Col     Rs      N_factor       ');
   writeln(lst,'Is          Vb         CC');
end;

procedure record_diode_parameter;
begin
   assign(number_transfer,buffer);
   reset(number_transfer);
   seek(number_transfer,filesize(number_transfer));
   clear_line;
   gotoxy(1,1); write('Enter row & col : '); readln(row,col);
   write(number_transfer,row,col,rs,nf,is,vb,gf);
   write(lst,row:2:0,'  ',col:2:0,'  ',rs:10,'  ',nf:10,'  ');
   writeln(lst,is:10,'  ',vb:10,'  ',gf:10);
   close(number_transfer);
end;

procedure write_diode_parameter;
begin
   write(lst,'Row  Col     Rs      N_factor       ');
   writeln(lst,'Is          Vb         CC');
   clear_line;
   gotoxy(1,1); write('Enter filename : '); readln(buffer);
   assign(number_transfer,buffer);
   reset(number_transfer);
   while not(eof(number_transfer)) do
   begin
      read(number_transfer,row,col,rs,nf,is,vb,gf);
```

```
      write(lst,row:2:0,'  ',col:2:0,'  ',rs:10,'  ');
      writeln(lst,nf:10,'  ',is:10,'  ',vb:10,'  ',gf:10);
   end;
   close(number_transfer);
end;

begin  {main program}
ylstart := -7.0;    ylstop := -2.0;
xstart := 0;        xstop := 2.0;
area := 18e-8;    {um e-8}
main_menu:
   clrscr;
   textmode(bw80);
   clear_line;
   gotoxy(1,1); write('Main Menu>> (t)test-iv (r)read-iv (q)quit : ');
loop1:
   if not(keypressed) then goto loop1
   else read(kbd, main_mode);
   case main_mode of
      't' :  test(v, i, ndata);
      'r' :  begin
                readfile(v, i, ndata);
                plot_ivdata(v, i, ndata);
                curve_fit_ivdata(v, i, ndata, rs, nf, gf, is);
             end;
      'q' :  goto quit;
   end;
   find_barrier_height;
   plot_ivcurve;
sub_menu:
   clear_line;
   gotoxy(1,1); write('Sub Menu>> ');
   write('(d)disp (l)lst (s)sav (b)buf r(rec) w(wri) q(qit) : ');
loop2:
   if not(keypressed) then goto loop2
   else read(kbd, sub_mode);
   case sub_mode of
      'd' : display(v, i, ndata);
      'l' : list(v, i, ndata);
      's' : writefile(v, i, ndata);
      'q' : goto main_menu;
      'b' : create_new_buffer;
      'r' : record_diode_parameter;
      'w' : write_diode_parameter;
   end;
   goto sub_menu;
quit:
end.
```

Appendix E

Computer Program Listing for Doping Profile Measurement

```
program cv_measurement_of_Schottky_diode;


{       The name of this program is cvdiode. It was developed
    for measuring the C-V characteristics and to calculate the
    doping profile of a diode. The program is written in
    Turbo Pascal. An IBM-XT is used to control the Hp4280A
    C-meter, whose address is 17. The measured data can be saved
    on the disk, retrieved from the disk, displayed on the monitor,
    or listed on the printer. Once a command is entered, press
    return to invoke the command. Default value for the diode
    area is 2.6e-3 cm^2, which is the area of the mercury
    probe. The doping concentration ranges from 10^16 cm^-3 to
    10^18 cm^-3, and the depth ranges from 0 um to 0.5 um. The
    capacitance ranges from 80 pF to 400 pF, and the voltage ranges
    from -5 V to 0.5 V at 0.5 V step increment. The duffusion barrier
    potential is assumed to be 0.94 V for aluminum on GaAs with
    about 10^17 cm^-3 doping concentration. The zero bias
    capacitance (Co), capacitance ratio (Cr), and capacitance
    exponent (Gm) are determined from curve-fitting the measured
    C-V characteristic. The correlation coefficient (cc) for this fit
    is also displayed on the monitor. The measured doping profile is
    least-square-fitted with the hyperabrupt doping function.     }




type
   sdesc = record
           len : byte;
           addr : integer;
           end;
   datarray = array[1..100] of real;

const
   xmin = 190; xmax = 550;
   ymin = 33;  ymax = 163;

label
   quit, loop_1, loop_2;

var
   cmd, recv : string[200];
   vstart, vstop, vstep : string[18];
   cmddesc, recvdesc : sdesc;
   my_address, system_controller, len, status : integer;
   quote : char;
   i, n, first, nchar, code: integer;
```

```
   cap, cond, volt, buffer :string[20];
   capacitance, conductance, voltage : real;
   slope_a, gamma, intercept_b, correlation_coeff : real;
   c, v, g, lnv, lnc, lnxd, lnnd, nd, xd : datarray;
   eo, er, eq, ci, area, dcdv, nd1, nd2, nd3, cratio : real;
   co, nn, no, phi : real;
   data_transfer : text;
   filename : string[8];
   ylstart, ylstop, ylunit, xstart, xstop, xstep, xunit : real;
   ndata : integer;
   q, cont : char;
   number_transfer : file of real;
   row, col : real;

procedure initialize(var addr, level : integer);
   external 't488init';

procedure transmit(var s : sdesc; var status : integer);
   external 't488xmit';

procedure receive(var r : sdesc; var len, status : integer);
   external 't488recv';

function ten_to_power(n : integer) : real;
begin
   ten_to_power := exp( n*ln(10) );
end; {of ten_to_power}

function ten_to(n : real) : real;
begin
   ten_to := exp( n*ln(10) );
end; {of ten_to}

procedure clear_line;
begin
gotoxy(1,1); write('                                             ');
             write('                                             ');
end; {clear_line}

procedure convert_lin_to_log(var z : datarray; ndata : integer; q : char);
var
   i : integer;
begin
   clear_line;
   gotoxy(1,1); write('taking log(',q,') ... ');
   for i := 1 to ndata do begin
      z[i] := ln(z[i])/ln(10.0);
   end;
   write('done');
   read(q);
end; {of convert_lin_to_log}

procedure lsf(ndata : integer; x, y : datarray;
```

```
                var slope, intercept, cc : real);
var
   xsum, ysum, xysum, xssum, yssum, xden, yden, rden : real;
begin
   xsum := 0; ysum := 0; xysum := 0;
   xssum := 0; yssum := 0;
   for i := 1 to ndata do begin
     xsum := xsum + x[i];
     ysum := ysum + y[i];
     xysum := xysum + x[i]*y[i];
     xssum := xssum + x[i]*x[i];
     yssum := yssum + y[i]*y[i];
   end;
   xden := ndata*xssum-xsum*xsum;
   yden := ndata*yssum-ysum*ysum;
   rden := sqrt(xden*yden);
   slope := (ndata*xysum-xsum*ysum)/xden;
   intercept := (ysum*xssum-xysum*xsum)/xden;
   cc := (ndata*xysum-xsum*ysum)/rden;
end; {of lsf}

procedure draw_box (x1,y1,x2,y2,color : integer);
begin
   draw (x1,y1,x1,y2,1);
   draw (x1,y2,x2,y2,1);
   draw (x2,y2,x2,y1,1);
   draw (x2,y1,x1,y1,1);
end; {of draw_box}

procedure draw_x_ticks(x1,y1,x_increment,x2 : integer);
begin
   while (x1<=x2) do begin
   draw (x1,(y1+1),x1,(y1-2),1);
   x1 := x1 + x_increment;
   end; {while}
end; {of draw_x_ticks}

procedure draw_y_ticks(x1,y1,y_increment,y2 : integer);
begin
   while (y1<=y2) do begin
   draw ((x1+2),y1,(x1-4),y1,1);
   y1 := y1 + y_increment;
   end; {while}
end; {of draw_y_ticks}

procedure draw_graph(x1,y1,x2,y2,
   x_increment, y_increment_left, y_increment_right :integer);
begin
   draw_box(x1,y1-2,x2,y2+2,1);
   draw_x_ticks (x1,y1,x_increment,x2);
   draw_x_ticks (x1,y2,x_increment,x2);
   draw_y_ticks (x1,y1,y_increment_left,y2);
   draw_y_ticks (x2,y1,y_increment_right,y2);
```

```
end; {of procedure draw_graph}

procedure write_x_coordinates(x1,y1,x_increment :integer;
   x1c,xc_increment,x2c : real; field,fix :integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (x1c:field:fix);
   x1c := x1c + xc_increment;
   x1 := x1 + x_increment;
   until x1c > x2c;
end; {of write_x_coordinates}

procedure write_y_coordinates(x1,y1,y_increment :integer;
   y1c,yc_increment,y2c : real; field, fix : integer);
begin
   repeat
   gotoxy((x1 div 8), (y1 div 8));
   write (y1c:field:fix);
   y1c := y1c + yc_increment;
   y1 := y1 + y_increment;
   until y1c < y2c;
end; {of write_y_coordinates}

procedure set_up;
begin
      hires; hirescolor(15);
      draw_graph(xmin,ymin,xmax,ymax,(xmax-xmin)div 5,
                 (ymax-ymin)div 5, (ymax-ymin)div 5);
end; {of set_up}

procedure write_y_axis;
begin
   write_y_coordinates(xmin-40,ymin+7,(ymax-ymin) div 5,
                       ylstop,(ylstart-ylstop)/5,ylstart,4,1);
end; {of write_y_axis}

procedure write_x_axis;

begin
   write_x_coordinates (xmin,ymax+14,(xmax-xmin) div 5,
                        xstart,(xstop-xstart)/5,xstop,4,2);
end; {of write_x_axis}

procedure write_y_axis_exp;
begin
   write_y_coordinates(xmin-40,ymin+7,(ymax-ymin) div 2,
                       ylstop,(ylstart-ylstop)/2,ylstart,4,1);
end; {of write_y_axis_exp}

procedure plot_a_pixel(x, y : real);
var
   xcor, ycor : integer;
```

```
begin
   xcor := round((x-xstart)*(xmax-xmin)/(xstop-xstart));
   ycor := round((ylstop-y)*(ymax-ymin)/(ylstop-ylstart));
   graphwindow(xmin,ymin,xmax,ymax);
   plot(xcor,ycor,1);
end;

procedure plot_a_point(x, y : real);
var
   xcor, ycor : integer;
begin
   xcor := round((x-xstart)*(xmax-xmin)/(xstop-xstart));
   ycor := round((ylstop-y)*(ymax-ymin)/(ylstop-ylstart));
   graphwindow(xmin,ymin,xmax,ymax);
   plot(xcor-1,ycor,1);
   plot(xcor+1,ycor,1);
   plot(xcor,ycor-1,1);
   plot(xcor,ycor+1,1);
end; {plot_a_point}

procedure plot_datafile(x, y : datarray; ndata : integer);
var
  i, xcor, ycor, yd : integer;
  amp, l, xc : real;
  q : char;

begin
   set_up;
   gotoxy(1,1); write('enter ylstart ylstop xstart xstop : ');
   readln(ylstart, ylstop, xstart, xstop);
   write_y_axis; write_x_axis;
   for i := 1 to ndata do begin
      xcor := round((x[i]-xstart)*(xmax-xmin)/(xstop-xstart));
      ycor := round((ylstop-y[i])*(ymax-ymin)/(ylstop-ylstart));
      graphwindow(xmin,ymin,xmax,ymax);
      plot(xcor-1,ycor,1);
      plot(xcor+1,ycor,1);
      plot(xcor,ycor-1,1);
      plot(xcor,ycor+1,1);
   end;
end; {of plot_datafile}

procedure readfile(var x, y : datarray; var ndata : integer);
var
   i : integer;
   data_transfer : text;
begin
   clear_line;
   gotoxy(1,1); write('enter filename(read) : ');
   readln(filename);
   assign(data_transfer,filename);
   ndata := 0;
   reset(data_transfer);
```

```
   while not eof(data_transfer) do begin
     ndata := ndata + 1;
     readln(data_transfer,x[ndata],y[ndata]);
   end;
   close(data_transfer);
end; {of readfile}

procedure writefile(x, y : datarray; ndata : integer);
var
   i : integer;
   data_transfer : text;
begin
   clear_line;
   gotoxy(1,1); write('enter filename(write) : ');
   readln(filename);
   assign(data_transfer,filename);
   rewrite(data_transfer);
   for i := 1 to ndata do
   begin
      writeln(data_transfer,x[i],'  ',y[i]);
   end;
   close(data_transfer);
end; {of writefile}

procedure list(x, y : datarray; ndata : integer);
var
   i : integer;
begin
   for i := 1 to ndata do begin
      writeln(lst,x[i],'  ',y[i]);
   end;
end; {of writedata}

procedure display(x, y : datarray; ndata : integer);
var
   i : integer;
   q : char;
begin
   clear_line;
   for i := 1 to ndata do begin
      gotoxy(1,1); writeln(ndata,' -  ',i,' ',x[i],'  ',y[i]);
   read(q);
   end;
end; {of display}

procedure create_new_buffer;
var
   select : integer;
begin
   clear_line;
   gotoxy(1,1); write('Enter 1 for create & 0 for append to a file : ');
   read(select);
   clear_line;
```

```
   gotoxy(1,1); write('Enter filename : '); readln(buffer);
   if select = 1 then
   begin
   assign(number_transfer,buffer);
   rewrite(number_transfer);
   close(number_transfer);
   end;
   write(lst,'Row  Col    C.5        Co');
   writeln(lst,'Cm1      Cm2        Vb      Gamma');
end;

procedure record_diode_parameter;
var
   vb, cp5, co, cm1, cm2 : real;
begin
   assign(number_transfer,buffer);
   reset(number_transfer);
   seek(number_transfer,filesize(number_transfer));
   clear_line;
   gotoxy(1,1); write('Enter row & col : '); readln(row,col);
   vb := xstart;
   cp5 := c[ndata];
   co := c[ndata-1];
   cm1 := c[ndata-2];
   cm2 := c[1];
   write(number_transfer,row,col,cp5,co,cm1,cm2,vb,gamma);
   write(lst,row:2:0,'  ',col:2:0,'  ',cp5:9,'  ');
   writeln(lst,co:9,'  ',cm1:9,'  ',cm2:9,'  ',vb:3:1,'  ',gamma:9);
   close(number_transfer);
end;

procedure write_diode_parameters;
var
  vb, cp5, cp0, cm1, cm2, gm : real;
begin
   write(lst,'Row  Col   C.5        Co        ');
   writeln(lst,'Cm1       Cm2          Vb      Gamma');
   clear_line;
   gotoxy(1,1); write('Enter filename : '); readln(buffer);
   assign(number_transfer,buffer);
   reset(number_transfer);
   while not(eof(number_transfer)) do
   begin
      read(number_transfer,row,col,cp5,cp0,cm1,cm2,vb,gm);
      write(lst,row:2:0,'  ',col:2:0,'  ',cp5:9,'  ');
      writeln(lst,cp0:9,'  ',cm1:9,'  ',cm2:9,'  ',vb:9,'  ',gm:9);
   end;
   close(number_transfer);
end;

procedure cvtest(var v, c : datarray; var ndata : integer);
label
   again, loop;
```

```
var
   cf, vf, vinc : real;
begin
again:
   set_up;
   ylstart := 80.0;   ylstop := 400.0;  ylunit := 1e-12;
   xstart := -5.0;    xstop := 0.5;     xstep := 0.5;
   gotoxy(1,1);
   write('Cd_vs_Vd>> ylstart ylstop ylunit xstart xstop xstep : ');
   readln(ylstart, ylstop, ylunit, xstart, xstop, xstep);
   ndata := trunc( ((xstop-xstart) / xstep) ) + 1;
   write_y_axis;
   write_x_axis;
   quote := chr(39);
   cmddesc.addr := ofs(cmd) + 1;
   recvdesc.addr := ofs(recv) + 1;

   my_address := 21;
   system_controller := 0;
   initialize(my_address, system_controller);

   cmd := 'IFC REN MTA LISTEN 17' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'DATA ' + quote + 'LE2 Z0' + quote + '13 10' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   str(xstart,vstart);
   str(xstop,vstop);
   str(xstep,vstep);
   cmd := 'DATA ' + quote + 'IB2, CE1, PS' + vstart + ', PP' + vstop +
          ', PE' + vstep + ', PL1, PD1, TR3, SW1' + quote + '13 10' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   cmd := 'TALK 17' ;
   cmddesc.len := length(cmd);
   transmit(cmddesc, status);

   recv := '                                                                 ';
   recvdesc.len := length(recv);
   receive(recvdesc,len,status);
   for i := 1 to ndata do begin
      recv := '                                                              ';
      recvdesc.len := length(recv);
      receive(recvdesc,len,status);
      first := pos('C',recv) + 2;
      nchar := pos(',',recv) - first;
      cap := copy(recv,first+1,nchar);
      delete(recv,1,pos(',',recv));
      first := pos('G',recv) + 2;
```

```
      nchar := pos(',',recv) - first;
      cond := copy(recv,first+1,nchar);
      delete(recv,1,pos(',',recv)+1);
      volt := recv;
      if recv[1] = '+' then delete(volt,1,1);
      val(cap,capacitance,code);
      val(cond,conductance,code);
      val(volt,voltage,code);
      c[i] := capacitance;
      v[i] := voltage;
      g[i] := conductance;
      plot_a_point(v[i], c[i]/ylunit);
   end;
   {if v[i] is replace with v[i]-phi, fit will be better}
   for i := 1 to ndata do begin
      lnc[i] := ln(c[i])/ln(10);
      lnv[i] := ln(1-(v[i]/phi))/ln(10);
   end;
   slope_a := 0;
   intercept_b := 0;
   correlation_coeff := 0;
   lsf(ndata, lnv, lnc, slope_a, intercept_b, correlation_coeff);
   gamma := -slope_a;
   co := ten_to(intercept_b);
   cratio := c[ndata]/c[1];
   gotoxy(1,5); writeln('Cn = ',c[ndata]:8);
   gotoxy(1,6); writeln('C1 = ',c[1]:8);
   gotoxy(1,7); writeln('CR = ',cratio:2:1);
   gotoxy(1,10); writeln('Co = ',co:8);
   gotoxy(1,11); writeln('Gm = ',gamma:4:3);
   gotoxy(1,12); writeln('cc = ',correlation_coeff:6:5);
   vinc := (xstop-xstart)/100.0;
   vf := xstart;
   repeat
      cf := co / exp( gamma*(ln(1-vf/phi)) );
      plot_a_pixel(vf, cf/ylunit);
      vf := vf + vinc;
   until (vf = xstop) or (vf > xstop);
loop:
   clear_line;
   gotoxy(1,1); write('Cd_vs_Vd>> (s)sav (d)disp ');
   write(' (l)lst (r)rep (w) wri (c) crt (k) kip (q) n_x : '); read(q);
   case q of
      's' : writefile(v, c, ndata);
      'd' : writefile(v, c, ndata);
      'l' : list(v, c, ndata);
      'r' : goto again;
      'w' : write_diode_parameters;
      'c' : create_new_buffer;
      'k' : record_diode_parameter;
   end;
   if q <> 'q' then goto loop;
end; {of cvtest}
```

```
procedure read_cv(var x, y : datarray; var ndata : integer);
label
   again, loop;
var
   i : integer;
   cf, vf, vinc : real;
begin
again:
   readfile(x, y, ndata);
   display(x, y, ndata);
   set_up;
   ylstart := 80.0;   ylstop := 400.0;  ylunit := 1e-12;
   xstart := -5.0;    xstop := 0.5;
   gotoxy(1,1); write('Cd_vs_Xd>> ylstart ylstop ylunit xstart xstop : ');
   readln(ylstart, ylstop, ylunit, xstart, xstop);
   write_y_axis;
   write_x_axis;
   for i := 1 to ndata do begin
      plot_a_point(v[i],c[i]/ylunit);
   end;
   {if v[i] is replace with v[i]-phi, fit will be better}
   for i := 1 to ndata do begin
      lnc[i] := ln(c[i])/ln(10);
      lnv[i] := ln(1-(v[i]/phi))/ln(10);
   end;
   slope_a := 0;
   intercept_b := 0;
   correlation_coeff := 0;
   lsf(ndata, lnv, lnc, slope_a, intercept_b, correlation_coeff);
   gamma := -slope_a;
   co := ten_to(intercept_b);
   cratio := c[ndata]/c[1];
   gotoxy(1,5); writeln('Cn = ',c[ndata]:8);
   gotoxy(1,6); writeln('C1 = ',c[1]:8);
   gotoxy(1,7); writeln('CR = ',cratio:3:2);
   gotoxy(1,10); writeln('Co = ',co:8);
   gotoxy(1,11); writeln('Gm = ',gamma:4:3);
   gotoxy(1,12); writeln('cc = ',correlation_coeff:6:5);
   vinc := (xstop-xstart)/100.0;
   vf := xstart;
   repeat
      cf := co / exp( gamma*(ln(1-vf/phi)) );
      plot_a_pixel(vf, cf/ylunit);
      vf := vf + vinc;
   until (vf = xstop) or (vf > xstop);
loop:
   clear_line;
   gotoxy(1,1); write('Cd_vs_Vd>> (s)save (d)display ');
                write('(l)list (r) repeat (q)quit : '); read(q);
   if q = 's' then writefile(v, c, ndata);
   if q = 'd' then display(v, c, ndata);
   if q = 'l' then list(v, c, ndata);
```

```
   if q = 'r' then goto again;
   if q <> 'q' then goto loop
end;

procedure get_plot_a_file;
label
   again, loop;
var
   x, y : datarray;
   ndat : integer;
   q : char;

begin
again:
   readfile(x, y, ndat);
   display(x, y, ndat);
   plot_datafile(x, y, ndat);
loop:
   clear_line;
   gotoxy(1,1); write('Get & Plot>> (d)display ');
                write('(l)list (r) repeat (q)quit : '); read(q);
   if q = 'd' then display(x, y, ndat);
   if q = 'l' then list(x, y, ndat);
   if q = 'r' then goto again;
   if q <> 'q' then goto loop;
end;

begin {main}
loop_1:
   clrscr;
   textmode(bw80);
   phi := 0.94;  {phi varies slightly with doping concentration}
   gotoxy(1,1);
   write('Menu>> m(measure-cv) r(read-cv) p(get&plot) (q)quit : ');
   read(q);
   case q of
      'm' : cvtest(v, c, ndata);
      'r' : read_cv(v, c, ndata);
      'p' : get_plot_a_file;
      'q' : goto quit;
   end;
   clrscr;
   if q = 'p' then goto loop_1;
   set_up;
   ylstart := 16.0;   ylstop := 18.0;
   xstart := 0.0;    xstop := 0.5;    xunit := 1.0;
   gotoxy(1,1);
   write('Nd_vs_Xd>> ylstart, ylstop, xstart, xstop, xunit [exp & um] :');
   readln(ylstart, ylstop, xstart, xstop, xunit);
   write_y_axis_exp;
   write_x_axis;
   eo := 8.85e-14;
   er := 13.1;
```

```
    eq := 1.6e-19;
    area := 2.6e-3; {area of Hg probe given in manual = 2.6e-3 [cm2]}
    clear_line;
    gotoxy(1,1); write('Nd_vs_Xd>> diode area [cm2] : ');
    readln(area);
    for i := 1 to ndata-1 do begin
       ci := ( c[i] + c[i+1] )/2;
       xd[i] := 1e4 * eo * er * area / ci;   {in units of microns}
       dcdv := ( c[i+1]-c[i] ) / ( v[i+1]-v[i] );
       nd1 := ci/eq;                  {nd1 := ci * ci * ci;}
       nd2 := ci/(eo*er);             {nd2 := eq * eo *er;}
       nd1 := nd1 * nd2;              {nd3 := area * area * dcdv;}
       nd1 := nd1 * ci / area;        {nd1 := nd1/nd2;}
       nd[i] := nd1/(area * dcdv);
       {nd[i] := nd1/nd3; units of atoms per cm-3}
       lnnd[i] := ln(nd[i])/ln(10.0);
       lnxd[i] := ln(xd[i])/ln(10.0);
       plot_a_point(xd[i]/xunit,lnnd[i]);
    end;
    slope_a := 0;
    intercept_b := 0;
    correlation_coeff := 0;
    lsf(ndata-1, lnxd, lnnd, slope_a, intercept_b, correlation_coeff);
    nn := -slope_a;
    no := ten_to(intercept_b);
    gotoxy(1,6); writeln('No = ',no:8);
    gotoxy(1,7); writeln('n  = ',nn:4:3);
    gotoxy(1,8); writeln('cc = ',correlation_coeff:6:5);

loop_2:
    clear_line;
    gotoxy(1,1);
    write('Nd_vs_Xd>> (s)save (d)display (l)list (q)quit: '); read(q);
    if q = 's' then writefile(xd, nd, ndata-1);
    if q = 'd' then display(xd, nd, ndata-1);
    if q = 'l' then list(xd, nd, ndata-1);
    if q <> 'q' then goto loop_2;
    goto loop_1;
quit:
end.
```