AUTOMATED MALFUNCTION DIAGNOSIS OF INTEGRATED CIRCUIT MANUFACTURING EQUIPMENT

by

Gary Stephen May

Memorandum No. UCB/ERL M91/33

26 April 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

AUTOMATED MALFUNCTION DIAGNOSIS OF INTEGRATED CIRCUIT MANUFACTURING EQUIPMENT

by

Gary Stephen May

Memorandum No. UCB/ERL M91/33

26 April 1991

AUTOMATED MALFUNCTION DIAGNOSIS OF INTEGRATED CIRCUIT MANUFACTURING EQUIPMENT

by

Gary Stephen May

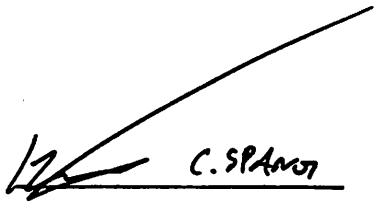
ABSTRACT

Manufacturing quality products in an integrated circuit fabrication facility requires that thousands of individual process variables be strictly controlled. Although in-line measurements and electrical test data have historically been used to detect process fluctuations, these methods alone have become inadequate for rapidly identifying possible problems in processes with a very narrow range of acceptable performance.

Individual process steps are conducted by complex pieces of fabrication equipment. When unreliable performance causes this equipment to vary beyond desired limits, overall product quality is jeopardized. Since process shifts resulting from faulty equipment can degrade semiconductor products to an unacceptable level, it is essential that root causes for the malfunctions be diagnosed and corrected quickly to prevent the continued occurrence of expensive misprocessing. However, with the advent of highly proficient sensors designed to monitor process conditions *in-situ*, it has become feasible to perform such malfunction diagnosis on a real-time basis. Therefore, methods of equipment diagnosis which utilize these capabilities are critical to the overall success of the semiconductor production process.

This dissertation presents a general methodology for the automated diagnosis of integrated circuit fabrication equipment. The technique presented combines the best aspects of quantitative algorithmic diagnosis and qualitative knowledge-based approaches. Evidence from equipment

maintenance history, real-time sensor data and in-line measurements are integrated using evidential reasoning techniques within the Berkeley Computer-Aided Manufacturing (BCAM) framework. This methodology is applied to the identification of faults in the Lam Research Autoetch 490 automated plasma etching system.

Signature: 

Committee Chairman

For correcting my homework when I was a child,

For building my spiritual base as I grew to adulthood,

For writing cherished words of encouragement after I left the nest,

For being an unlimited reservoir of support and love throughout my life,

To My Mother

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deepest appreciation and gratitude to my research advisor, Professor Costas J. Spanos for his insightful guidance and support throughout the duration of this study. I also thank Dean David A. Hodges and Professor Alice M. Agogino for serving on my dissertation committee. Furthermore, I thank Dean Hodges for his unwavering support and encouragement of CIM research at Berkeley. Finally, I am grateful to Professor Ping K. Ko for his advice during the attainment of my Master's degree.

I would be remiss if I did not acknowledge the efforts of J. Huang for her invaluable assistance in performing plasma etch experiments, Dr. N. Chang for his work in designing a prototype diagnostic system for LPCVD, H. Guo for her aid in equipment monitoring during experimentation, Dr. K. Lin for sharing his equipment modeling expertise, H. Liu for assistance in developing the BCAM user interface, and D. Mudie for his work on FAULTS, the Berkeley equipment record-keeping system.

Special thanks are also appropriate for the efforts of the staff of the Berkeley Microelectronics Laboratory, who significantly aided my knowledge of equipment operation, especially R. Norman. I would also like to specifically acknowledge B. Hamilton and K. Voros for their overall support of CIM research.

For many pleasurable experiences and the sharing of technical expertise, thanks to the remaining members of the CIM/CAM groups at Berkeley: Professor L. Rowe, Dr. Z.-M. Ling, Ms. L. Massa-Lochridge; and my student colleagues: S. Bana, B. Bombay, E. Boskin, R. Chen, C. Hegarty, S. Leang, S. Lee, T. Luan, D. Mudie, B. Smith, C. Wang and C. Williams. Special thanks are extended to E. Boskin and S. Lee for proofreading portions of this dissertation.

The presence of several fellow students has also been a key ingredient in my enjoyment of

graduate study at Berkeley. It is not possible to list all of their names here, but among them are C. Brooks, C. Coleman, J. Duster, L. Haynes, K. Kornegay, E. Page, C. Paris, I. Porche, J. Reason and V. Taylor. Dr. Sheila Humphreys and Genevieve Thiebaut have also been constant sources of assistance and encouragement.

Finally, I am eternally grateful to God and my family. Without their motivation, wisdom, sacrifice, and love, this dissertation could not have been accomplished.

This research has been jointly sponsored by the Semiconductor Research Corporation (SRC), MICRO, Harris Semiconductor, IBM, Intel, National Semiconductor, Philips/Signetics, Rockwell International, Siemens AG, Texas Instruments, AT&T Bell Laboratories, and the National Science Foundation under Grant No. MIP 8909095.

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

In order to keep pace with economic competition abroad as well as rapid technological advancement in fabrication techniques, the semiconductor industry in the United States is paying an increasing amount of attention to issues related to semiconductor manufacturing [1]. As a result, the current level of activity in computer-aided manufacturing (CAM) and computer integrated manufacturing (CIM) techniques today is reminiscent of that in computer-aided design (CAD) several years ago.

Maintaining product quality throughout an integrated circuit (IC) manufacturing facility requires the strict control of literally thousands of process variables. These variables serve as input and output parameters for hundreds of distinct process steps. Thus, monitoring these process steps requires the manipulation of vast amounts of data which exists in various formats, including:

1) work-in-process (WIP) information
2) equipment maintenance records
3) real-time process data
4) in-line physical and/or electrical measurement data
5) final electrical test data

The implementation of a well-designed CIM system that manages these five critical datasets as well as the design and control various processes is essential for maintaining a competitive edge in the industry.

Although in-line measurements and final electrical test data have historically been used to detect process fluctuations, these methods have become more and more inadequate for identifying all possible problems in processes with a very narrow range of acceptable performance [2]. As a consequence, process failures often remain undetected until the completion of what is often a very long and costly production cycle.

Individual IC process steps are conducted by sophisticated and expensive pieces of fabrication equipment. When unreliable equipment performance causes operating conditions to vary beyond their desired limits, overall product quality is severely jeopardized. Consequently, fast and accurate methods of equipment malfunction diagnosis are essential to the success of the semiconductor production process.

Traditional approaches to diagnosis typically take place at the conclusion of a sequence of several fabrication steps. Since integrated circuit wafers are manufactured in a batch mode, the detection of errors usually comes after a large number of expensive wafers have been misprocessed. This is extremely costly in terms of product yield. Therefore, from an economic point of view, diagnosis is more beneficial when it is conducted during each individual process step. This implies that greater emphasis must be placed on the diagnosis of specific pieces equipment rather than entire processes. Moreover, with the advent of highly proficient sensors to monitor process conditions *in-situ* [3-4], it has become possible to perform malfunction diagnosis on a real-time basis.

This dissertation presents a general methodology for the automated diagnosis of IC fabrication equipment. The technique presented combines the best characteristics of quantitative algorithmic diagnosis and qualitative experiential approaches. Evidence from equipment maintenance history, real-time sensor data and in-line measurements are integrated using Dempster's rules of evidential reasoning [5]. Within the Berkeley Computer-Aided Manufacturing (BCAM)

framework, this methodology is applied to the identification of faults in a single-wafer plasma etching system.

1.2 The Berkeley CAM Architecture

The objective of the BCAM system is to improve both the productivity and the quality of IC manufacturing. The approach utilized by BCAM is the design of a flexible architecture to link various software modules in an integrated system to support design, manufacturing and testing of IC processes. In this way, BCAM aims to support all aspects of equipment operation and process control in semiconductor manufacturing.

The BCAM group has identified several capabilities which contribute to the efficient operation of individual pieces of manufacturing equipment. Among these are: statistical process control (SPC), recipe generation, real-time monitoring, maintenance and record-keeping, modeling, and malfunction diagnosis. These capabilities share a number of basic resources, including numerical optimizers, statistical routines, relational databases and user interfaces. They are also very tightly coupled. For example, SPC is used to trigger an alarm which initiates malfunction diagnosis procedures, which in turn make use of equipment models to infer equipment faults. Therefore, the seamless integration of all of these capabilities is a necessity for any useful CIM system. A graphical depiction of the BCAM framework appears in Figure 1.1.

In its current implementation, BCAM is a workstation-based system which employs the object-oriented features of the CLOS [6], C and C++ [7] programming languages. In addition, it uses the X Window system and the INGRES [8] relational database. Direct communication with semiconductor manufacturing equipment is accomplished by means of the Semiconductor Equipment Communications Standard II (SECSII) protocol [9]. BCAM is designed to support inter-equipment control in work-cell configurations, and it is a part of the Berkeley Computer Integrated Manufacturing (BCIM) system.

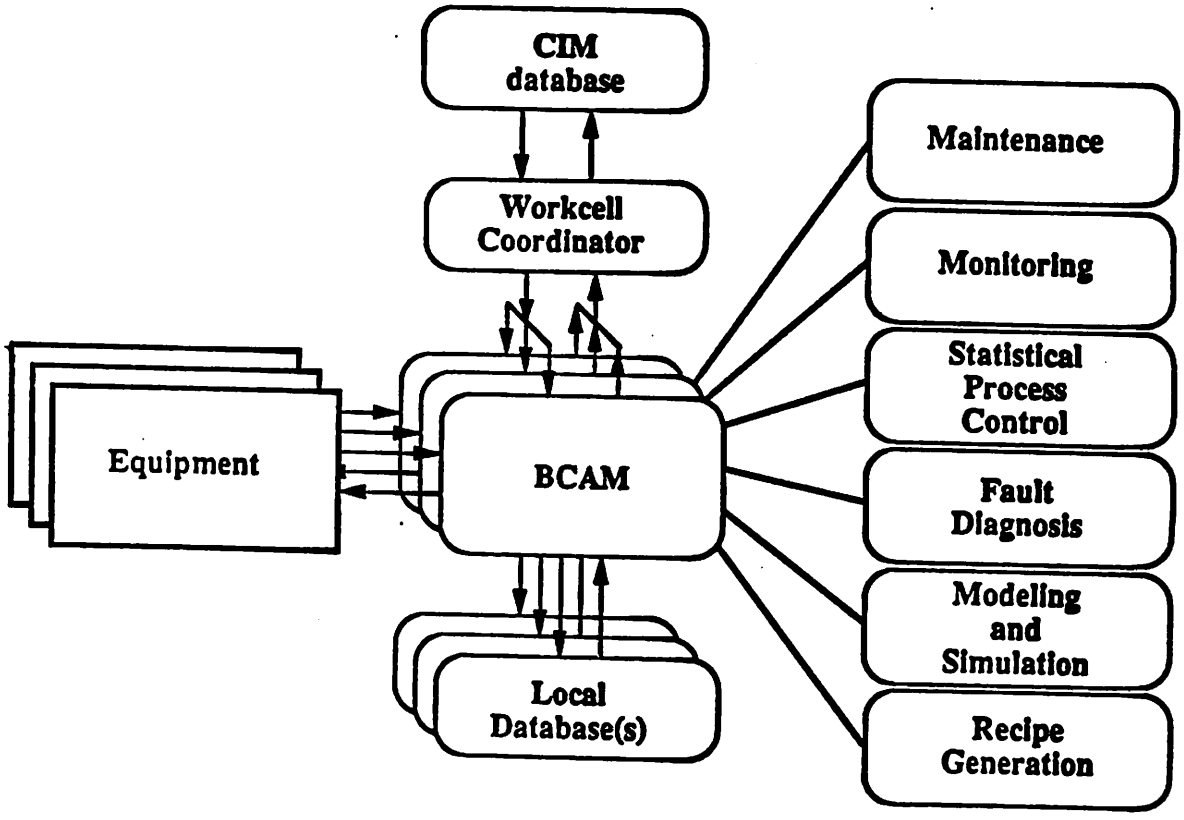


Figure 1.1 - The BCAM framework.

1.3 The Berkeley CIM System

In order to alleviate the cost and complications inherent in current CIM architectures, the Berkeley CIM group has proposed a two-level approach [1-2]. This two-level CIM architecture is depicted in Figure 1.2. The lower level includes the embedded workcell controllers which maintain the quality of processing equipment on a real-time basis. The upper level consists of a distributed network of multi-tasking workstations linked to a common relational database. The high-speed *CIMBUS* [2] provides communication among the various processors and acts as a coordinator among different applications.

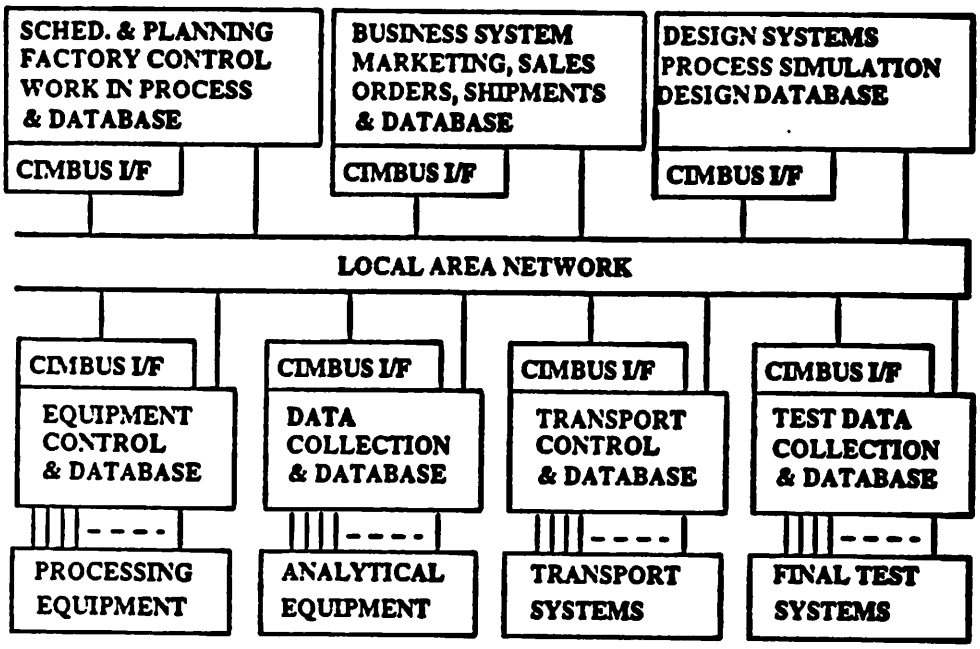


Figure 1.2 - Two-level CIM architecture.

Several recent technological advances have contributed greatly to this realization. First of all, it is now possible to constuct physically distributed, but logically integrated relational database systems whose ease of use reduces the effort required for both initial system development and subsequent modification. In addition, industry has widely accepted the use of high-bandwidth local area networks (LANs) that make it possible to connect process control applications directly to fabrication equipment in a cost-effective manner. These LANs enable the real-time monitoring that is critical to both SPC and equipment diagnosis. Finally, the emergence of artificial intelligence (AI) methodologies as an aid in automated decision-making, planning, scheduling, and diagnosis has eliminated some of the difficulties associated with these knowledge intensive and error-prone manufacturing activities.

The overall objective of the Berkeley CIM architecture is to develop software modules for controlling IC processing steps based upon the two-level model. This objective is also aided by the Berkeley Process Flow Language (BPFL) [10] and the Work-in-Progress (WIP) interpreter [11]. BPFL is used to describe manufacturing processes, and WIP executes the BPFL process specifications, handles equipment allocation and control, and collects and stores the data used to monitor the entire maunfacturing facility. BPFL and WIP operate at the upper level of the CIM architecture and send commands to the BCAM modules to initiate, monitor, diagnose, or generate a recipe for a process step. Using this architecture, it has become possible to implement a unified CIM system which is simpler and more flexible than earlier generation systems.

1.4 Thesis Organization

The subject of this dissertation is the development of an automated system to perform malfunction diagnosis on semiconductor manufacturing equipment within the overall BCAM framework. In particular, this system is applied to the diagnosis of a plasma etcher. A theoretical description of the inference technique employed to determine the equipment malfunctions is provided in Chapter 2. Chapter 3 contains an overview of the mechanical operation of the Lam Research Corporation Autoetch 490 plasma etcher, the particular machine to which the diagnostic scheme is applied. Experimental modeling of the etcher is described in Chapter 4. The approach taken to the generation and distribution of evidential support for fault hypotheses is discussed in Chapter 5. In Chapter 6, a detailed explanation of the software which implements the entire diagnostic system is provided. Finally, Chapter 7 presents a verification of the system by means of a few examples, as well as some overall conclusions and some suggestions for future work.

References for Chapter 1

[1] D. A. Hodges, L. A. Rowe and C. J. Spanos, "Computer Integrated Manufacturing," *International Electronic Manufacturing Technology Symposium*, September, 1989.

[2] N. H. Chang, "Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB/ERL M90/61*, May, 1990.

[3] S. B. Dolins, A. Srivastava, and B. E. Flinchbaugh, "Monitoring and Diagnosis of Plasma Etch Processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, no. 1, February, 1988.

[4] R. A. Gottscho and V. M. Donnelly, "Optical Emission Actinomerty and Spectral Line Shapes in RF Glow Discharges," *Journal of Applied Physics*, vol. 56, no. 2, July, 1984.

[5] G. Shafer, "A Mathematical Theory of Evidence," *Princeton University Press*, 1976.

[6] S. Keene, *Object Oriented Programming in Common Lisp*, New York: Addison-Wesley, 1989.

[7] K. Weiskamp and B. Flaming, *The Complete C++ Primer*, San Diego: Academic Press, 1990.

[8] *INGRES/Embedded SQL Companion Guide for C*, Relational Technologies, Inc., 1989.

[9] *The SECS Message Service for Bidirectional Transfer of SECS Messages, Part 2: Definition of Protocol*, SEMI Communication Committee, MAP/SECS Subnet, Semiconductor Equipment and Materials International, Jan. 1988.

[10] L. A. Rowe, C. B. Williams, and C. J. Hegarty, "The Design of the Berkeley Process-Flow Language," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB M90/62*, August, 1990.

[11] C. J. Hegarty, L. A. Rowe, and C. B. Williams, "The Berkeley Process-Flow Language WIP System," *Techon '90 Extended Abrstracts*, October, 1990.

CHAPTER 2

A METHODOLOGY FOR DIAGNOSTIC INFERENCE

2.1 Introduction

In semiconductor manufacturing equipment, a certain amount of inherent variability exists regardless of how well the machine is designed or maintained [1]. This "noise" is the result of numerous small and essentially uncontrollable causes. However, when this variability becomes large compared to background noise, significant performance shifts may occur. As an example of such a shift, consider the standard Shewhart control chart shown in Figure 2.1. This figure depicts a shift in the thickness of a particular thin film as integrated circuit wafers are processed in a fabrication line. Such process shifts are often indicative of equipment malfunctions. Since these shifts can degrade the overall fabrication process to an unacceptable level, it is critical that *assignable causes* for the equipment malfunctions be diagnosed and corrected quickly to prevent the continued occurrence of expensive misprocessing [2].

Several recent computerized diagnostic systems have had the objective of performing automated diagnosis of faults in both manufacturing processes and equipment. Algorithmic systems such as HIPPOCRATES [3] and MERLIN [4] have been developed to identify process faults from statistical inference procedures and electrical measurements performed on finished IC wafers. Although this technique makes good use of quantitative models of process behavior, it can only arrive at useful diagnostic conclusions in the limited regions of operation over which these models are valid. When catastrophic faults that destroy circuit functionality occur, these models can no longer adequately describe the failure mechanism [5]. Moreover, in critical process steps such as plasma etching, the theoretical basis for determining causal relationships is not very well understood (see Chapter 4), thereby limiting the usefulness of physical models [1,6-7].

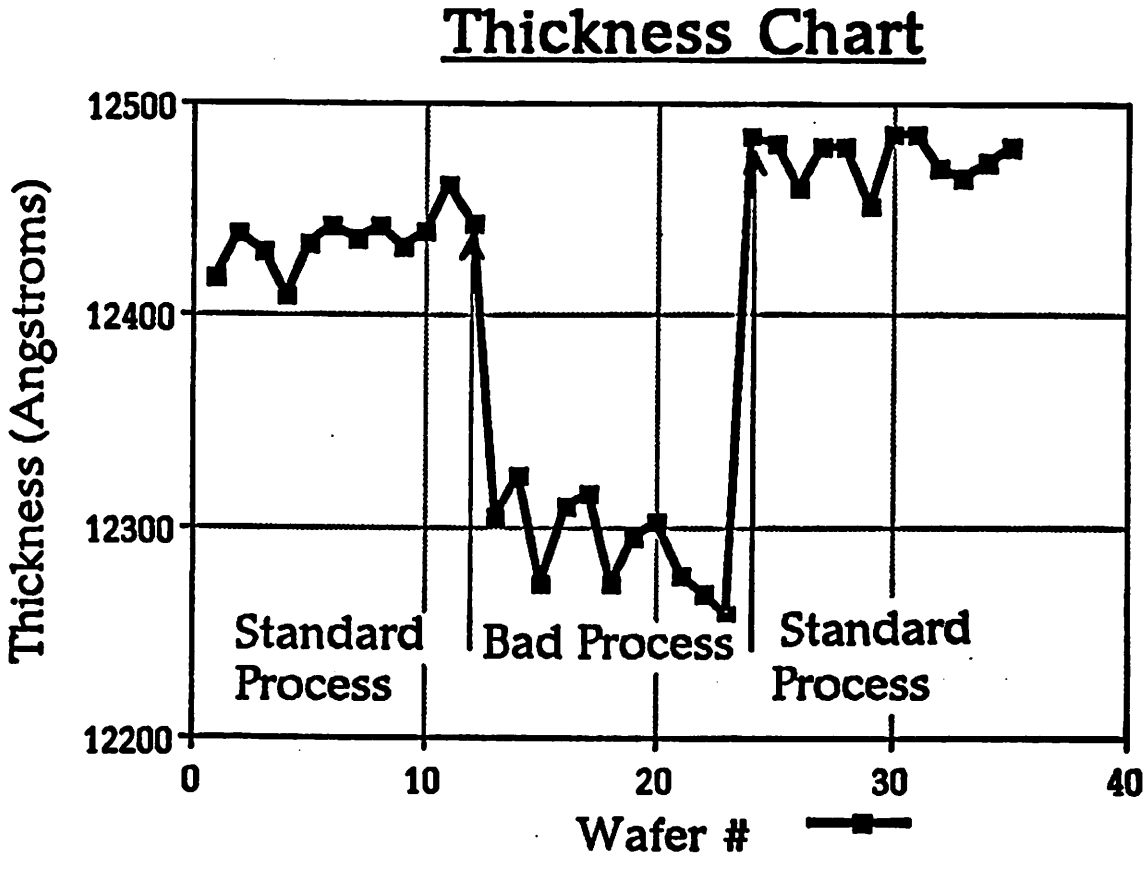


Figure 2.1 - Shewhart control chart illustrating a process shift.

Another algorithmic system has been designed to diagnose intra-wafer variability and spatial dependencies of process parameters [8]. This system utilizes *principal component analysis* [9] and pattern recognition techniques to determine the spatial distribution of process faults. Although the identification of the faults is accomplished through rigorous statistical means, the subsequent determination of their root causes is assisted by the quantification of empirical engineering knowledge. However, both this method and HIPPOCRATES perform diagnosis at the conclusion of IC processes. From a practical standpoint, this approach is inadequate since at that point in time, significant misprocessing and yield loss may have already taken place.

When attempting to diagnose unstructured problems which lack a solid conceptual foundation for reasoning, some success has been attained by approaches based upon quantifying expert knowledge. Expert systems such as PIES [10] and PEDX [11] are designed to draw upon experiential knowledge to develop qualitative models of process behavior. In this way, they are able to circumvent the difficulties encountered by algorithmic systems when quantitative relationships break down. Yet a purely knowledge-based approach often lacks the precision inherent in the deep-level physical models, and is thus incapable of deriving solutions for unanticipated situations from the underlying principles surrounding the process. Another shortcoming of purely expert diagnosis is its inability to identify concurrent multiple faults.

Both the algorithmic and the expert approaches to diagnosis outlined above typically take place following a sequence of several fabrication steps. However, since integrated circuits are manufactured in a "batch" mode, evidence pertaining to potential equipment malfunctions accumulates at irregular intervals throughout the process sequence. Often, diagnosis cannot be conclusive until subsequently acquired data verifies earlier hypotheses regarding the nature of the problem. Therefore, an essential requirement for a competent and effective diagnostic tool is a systematic methodology for combining evidence originating from numerous sources at random intervals.

The actual means by which the diagnostic system formulates fault hypotheses is known as the *inference engine* [12]. The inference engine performs the reasoning function for the entire system by systematically combining available evidence using a general methodology in conjunction with information contained in the *knowledge base* (see Chapter 3). This chapter presents an inference methodology for automated equipment diagnosis. Data from various sources is seamlessly integrated using Dempster's rules of evidential reasoning under uncertainty [13]. This technique integrates the best characteristics of quantitative algorithmic diagnosis and qualitative experiential approaches and has several advantages over other methods of inference. The

evidential reasoning approach has been implemented using *support logic programming* [14] in an object-oriented manner via the C++ programming language [15]. This methodology has been applied to fault identification in a plasma etcher.

2.2 Sources of Diagnostic Evidence

Evidence regarding fabrication equipment malfunctions originates from several sources, including equipment maintenance history, real-time sensor data, and physical in-line measurements of process parameters [16-17]. Both sensor and measurement data are constrained by first-principle physics and empirical relationships. Violation or significant deviation from these constraints is indicative of equipment faults. Each violated constraint may be subsequently mapped to a particular set of potential faults. As a simple example, consider the flow of mass through a pipe as shown in Figure 2.2. If F_1 represents the flow at the entry to the pipe and F_2 is the flow at its exit, then this system is constrained by the following expression relating the conservation of mass:

$$F_1 - F_2 = 0 \tag{2.1}$$

Flows F_1 and F_2 are monitored by sensors S_1 and S_2, respectively. The violation of constraint (2.1) in the positive direction might be diagnosed as either a leak or the failure of one of the two sensors, whereas a negative deviation would indicate a sensor failure only. In effect, diagnosis represents the logical combination of inferences drawn from a complete set of constraint equations. Such a complete set is referred to as the system of *governing equations* for a given process [18].

The solution of the governing equations for any given system falls into three distinct categories, depending upon the magnitude and direction of constraint deviation. This fact is illustrated through the following expressions:

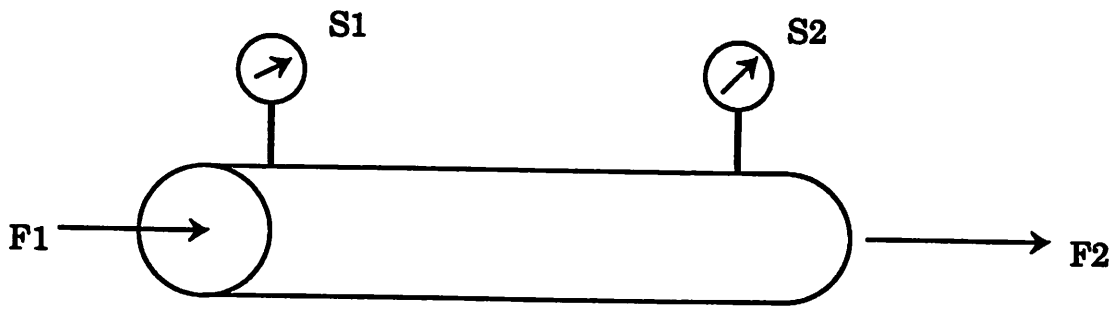


Figure 2.2 - Simple example of a physical constraint: the flow of mass through a pipe.

$$\begin{aligned} C^{+} &\equiv (C > tol) \\ C^{o} &\equiv (|C| \leq tol) \\ C^{-} &\equiv (C < tol) \end{aligned} \tag{2.2}$$

where C^{+}, C^{o} and C^{-} represent the conditions for positive constraint violation, constraint satisfaction, and negative constraint violation for a given equation C. The variable *tol* represents the tolerance over which a given constraint is permitted to vary. The analog to the simple pipe system in a plasma etcher is the mass flow controller (MFC) which controls the flow of gases into the process chamber [19]. In general, all manufacturing equipment is composed of numerous instances of systems, subsystems and components, each of which is influenced by particular faults which manifest themselves as observable symptoms that are detectable by sensors or in-line measurements.

2.3 A Survey of Existing Inference Methods

Each piece of evidence collected contributes incremental adjustments in belief regarding the

presence of a particular fault or fault group. In order to facilitate robust diagnosis, this information must be efficiently combined and integrated to produce a viable overall conclusion. Several methods have been proposed and investigated to achieve the necessary integration of evidence in this system. A brief survey of these approaches along with their relative merits is given below.

2.3.1 Boolean Logic

Classical logic requires the truth of any statement to be two-valued (either "true" or "false"). Reasoning systems based upon this approach draw an overall conclusion from a set of true and/or false statements by manipulating this binary information with classical Boolean algebra. However, the use of the Boolean approach for diagnostic applications is inadequate for several reasons. First, the assumption that all available information is binary in nature leads to misleading conclusions. This is due to the fact that data collected from manufacturing equipment is often uncertain. For example, consider the example of the flow of gas through a mass flow controller in the plasma etcher. In this situation, it is apparent that the accurate diagnosis of MFC faults depends heavily on the validity of the sensors which measure the gas flow. If a particular sensor is malfunctioning or slightly miscalibrated, the conclusions drawn about the process differ significantly from cases in which sensors are functioning properly. Classical Boolean logic lacks a method for attributing a degree of belief to sensor accuracy or to any other uncertain information.

In addition, the use of governing equations as constraints depends on the classification of violations into the categories C^+, C^o and C^-. An incorrect classification can drastically alter diagnosis by either eliminating otherwise viable hypotheses or by postulating spurious alternatives. At or near the threshold value tol, the inference is much too sensitive to incremental changes in the process conditions, thereby greatly exacerbating diagnostic instability in the presence of measurement noise [16-18].

Finally, classical knowledge bases are monotonic in nature. Statements regarding a given proposition lead to either the logical confirmation or contradiction of that proposition. Due to the complex interaction among various subsystems in IC fabrication equipment, this monotonic logical progression is not often the case. At one moment, a component failure due to overheating might be attributed to a mechanical fault in the cooling system. Later, however, further scrutiny of the available evidence might attribute the overheating to an insufficient supply of liquid coolant, thus repudiating the original hypothesis. Like fluctuations due to measurement noise, this constant state of flux in belief also renders classical reasoning techniques unstable.

2.3.2 Certainty Factors

Although the Boolean approach suffers from the shortcoming of two-valued logic, this inadequacy is overcome through the use of multivalued measures of certainty. In this way, the certainty of information may be expressed in terms of a "likelihood" rather than an exact truth. This allows diagnoses to be expressed in terms of ranked lists of fault possibilities, each with varying degrees of belief attached. These degrees of belief, known as *certainty factors*, are usually normalized to a standard scale such as the interval [-1,1]. Certainty factors, however, do not adequately address the notion of uncertainty since even the degree of certainty (or uncertainty) regarding a particular proposition will itself be uncertain. Despite the claims of the developers of diagnostic systems based on certainty factors that these factors possess an inherent advantage due to their intuitive "usability," studies have indicated that rule-based systems employing certainty factors are significantly less accurate than other techniques given uncertain evidence [20]. Moreover, it has been shown that particular classes of dependencies among uncertain beliefs cannot be represented by certainty factors in an efficient manner [21].

In addition, there has been much debate concerning the the development of a proper and rigorous methodology for combining certainty factors from multiple sources of evidence. There

is an implicit assumption in these systems that if a numerical certainty factor is attached to each premise in a knowledge base, then a certainty factor corresponding to a conclusion based on these premises may be expressed as a function of their individual certainty factors. This assumption is very questionable. In fact, it has been demonstrated that the original rules for evidence combination using certainty factors is ad hoc and lacks commutativity [22]. Since evidence is collected at irregular time intervals, commutativity is an essential property for accurate diagnosis. Although this discrepancy can be alleviated through the proper choice of a probabilistic interpretation for a given set of certainty factors under a given framework of inference, an infinite number of valid choices of interpretation exist. The question of which interpretation is optimal has yet to be conclusively answered.

Finally, a set of rules for factor combination under any probabilistic interpretation relies heavily upon the assumption of conditional independence of evidence for a given hypothesis [23]. This assumption is not justifiable in the equipment diagnosis regime. Due to the complex interaction between system components, the presence of one symptom often directly results in the occurrence of another (see Chapter 3). Referring to the previous example of the mass flow through a pipe, it would be erroneous to assume that the evidence collected by sensors S_1 and S_2 is independent. One possible scenario in which these two symptoms would be directly related would be in the event of an electrical systems failure which resulted in their mutual miscalibration. Thus, the symptoms related to electrical failure would have a direct impact on the symptoms which initiate the diagnosis of a possible leak.

2.3.3 Bayesian Inference

The Bayesian approach to reasoning involves the integration of both prior fault probabilities and conditional probabilities of faults given a certain set of symptoms. In general, given n *mutually exclusive* and *collectively exhaustive* events that define a state A, the conditional probability

(Pr) of a particular event B in this system is expressed as:

$$Pr(B|A_i) = \frac{Pr(A_i \cap B)}{Pr(A_i)} \tag{2.3}$$

where $i = 1,...,n$. In diagnosis, the events A_i could represent various equipment faults. The event B would then represent a particular symptom that occurs as a result of these faults.

The task of a diagnostic system is to determine the probability of fault f_i given that symptom s is present. This probability is also calculated using conditional probabilities as follows:

$$Pr(f_i|s) = \frac{Pr(f_i \cap s)}{Pr(s)} = \frac{Pr(s|f_i)Pr(f_i)}{Pr(s)} \tag{2.4}$$

Because faults f_i are collectively exhaustive, their union forms the entire set of possible faults, or:

$$F = f_1 + f_2 + \cdots + f_n \tag{2.5}$$

where F is the set of all possible faults. Set theory dictates that the intersection of a sum of sets is the sum of their intersections, thus:

$$(f_1 + f_2 + \cdots + f_n) = f_1 s + f_2 s + \cdots + f_n s \tag{2.6}$$

Furthermore, since faults f_i are mutually exclusive, their intersections are empty, or $(f_i s)(f_j s) = \varnothing$. Therefore, the probability of symptom s is the sum of the probabilities of all combinations $f_i s$:

$$Pr(s) = Pr(f_1 s) + Pr(f_2 s) + \cdots + Pr(f_n s) = \sum_1^n Pr(s|f_i)Pr(f_i) \tag{2.7}$$

Substituting (2.7) into (2.4) gives one version of Bayes' Theorem:

$$Pr(f_i|s) = \frac{Pr(s|f_i)Pr(f_i)}{\sum_1^n Pr(s|f_i)Pr(f_i)} \tag{2.8}$$

However, if one assumes conditional independence among individual symptoms s_i and s_j given that a particular fault f is known, then the computational complexity of (2.8) may be reduced using [18]:

$$Pr(f|S) = Pr(f) * \frac{Pr(s_1|f)}{Pr(s_1)} * \frac{Pr(s_2|f)}{Pr(s_2)} * \cdots \tag{2.9}$$

where $S = s_1, s_2, \cdots$. The assumption of conditional independence implies that if the true failure state is known, then the probability of one symptom (or sensor reading) is independent of another. In other words, for a known failure state, additional sensor readings provide no new information on the readings of another sensor.

The major difficulty in the implementation of the Bayesian approach is the large number of subjective probabilities which must be obtained (either from the domain expert or through experimentation) [23]. Since etch malfunction diagnosis entails over 40 distinct potentially faulty components (refer to Appendix 3.1) and about 15 relevant, observable symptoms (see Chapter 5), then over 615 probability values (600 conditional probabilities and 15 prior symptom probabilities) must be gathered in order to use equation (2.9). Moreover, if the possibility of multiple faults exists, this number becomes even higher. Determining such a huge number of probabilities can be both a time-consuming and expensive process.

However, even if all these probabilities could be obtained, the Bayesian approach requires that the faults themselves be mutually exclusive and collectively exhaustive. Since overlapping sets of faults are not at all uncommon, and due to the lack of available information regarding their mutual exclusivity, it can be difficult to model a system such that this assumption holds. Although efforts to circumvent these difficulties involving the use of so-called "belief networks" and influence diagrams [24] have had some degree of success, an efficient algorithm to solve the general inference problem using these networks has yet to be found.

2.3.4 Fuzzy Logic

Fuzzy logic utilizes the concept of *membership grade* to express variable degrees of membership of elements in imprecisely defined sets [12,25-26]. In this framework, a fuzzy set X can be described by a set of ordered pairs:

$$X = \left\{ (x_1,\zeta_1) , (x_2,\zeta_2) , \ldots \right\} \tag{2.10}$$

where ζ_i is a number in the interval [0,1] representing the grade of membership of x_i in X. Methodology exists which allows the standard set operations of union, intersection, and complementation on such fuzzy sets. Traditional logic is merely a subset of fuzzy logic in which each ζ_i is equal to one.

Since this paradigm introduces the formal concept of inexactness of information in a fairly rigorous manner, it is a natural means of dealing with the troubling presence of uncertainty in evidence. Although this methodology is ideal for enhancing knowledge base consistency through the unambiguous manipulation of inexact quantifiers used in natural languages (such as "almost", "very", etc.) [21], this type of formalism is most useful for expert systems with extensive user interfaces. However, as will be shown later, the diagnostic system which has been designed for the plasma etching application has little need for such semantic interpretation due to its use of numerically-based heuristics for belief generation and propagation (see Chapter 5).

2.3.5 Dempster-Shafer Theory

The methods of evidential reasoning outlined in Dempster-Shafer theory [13] appear to represent the most suitable compromise between the various disadvantages of the inference methods mentioned above. This technique allows the combination of various pieces of uncertain evidence obtained at irregular intervals. These evidence sources are usually assumed to be independent, but proper modifications can allow this requirement to be relaxed under certain

conditions (see § 2.5). Further, the preferred implementation of this approach results in continuously varying, non-monotonic belief functions which reflect the status of diagnostic conclusions at any given point in time. These belief functions can be chosen so that they are insensitive to measurement noise [16-17]. Dempster-Shafer theory provides conceptually straightforward reasoning and reduces to both the Boolean and Bayesian model in the limit when evidence takes on certain forms [27].

According to the basic tenets of Dempster-Shafer theory, the likelihood of proposition A is expressed as an interval $[s(A), p(A)]$ which lies in [0,1]. The parameter $s(A)$ represents the belief in A due to the amount of supporting evidence available, while $p(A)$ indicates the plausibility of A. Plausibility is defined as the degree to which contradictory evidence is lacking. In other words:

$$p(A) = 1 - s(\bar{A}) \tag{2.11}$$

where $s(\bar{A})$ is the measure of belief in the proposition $\bar{A}$. Therefore, the probability of the proposition A is bounded by $s(A)$ and $p(A)$. The quantity $u(A) = p(A) - s(A)$ is the uncertainty of A. For example, A [0.2,0.6] indicates that the probability of A is between 0.2 and 0.6 with an uncertainty of 0.4. If $u(A)$ is zero for all propositions in the knowledge base, then a more conventional system results [13].

Propositions in this arrangement are subsets of the set of all relevant hypotheses, θ. This superset is known as the *frame of discernment.* For the purposes of diagnosis, the frame of discernment is the union of all possible faults (hypotheses). Each piece of collected evidence can be mapped to a fault or group of faults which is a subset of θ. Evidential intervals for individual faults $[s(A), p(A)]$ are derived from a *basic probability mass distribution* (BPMD). The BPMD distributes numerical belief over the set of propositions in the fault hypothesis space domain. For instance, the BPM $m\langle A \rangle$ represents the sum of the belief attributed to A. Any residual belief in

the frame of discernment that cannot be attributed to any subset of θ is assigned directly to θ itself. This effectively serves to introduce uncertainty into diagnosis.

Using this framework, the belief and plausibility of a set of propositions A are given by:

$$s(A) = \Sigma m_i \langle A_i \rangle \tag{2.12}$$

$$p(A) = 1 - \Sigma m_i \langle B_i \rangle \tag{2.13}$$

where $A_i \subseteq A$ and $B_i \subseteq \bar{A}$. Thus, the total belief in A is the sum of beliefs ascribed to A and all subsets thereof.

2.4 Evidence Combination Using the Dempster-Shafer Methodology

In any given diagnostic application, there may be numerous sources of evidence contributing varying degrees of belief to several propositions under a common frame of discernment. Dempster's rule of combination [13,28] provides a deterministic and unambiguous method of combining BPMDs from separate and distinct sources of knowledge. Since the method is commutative, the order in which evidence is combined has no bearing on diagnostic conclusions. Further, as will be shown later, this rule is easily stated in a form which is readily programmable in a computer environment.

The rule for combining the observed basic probability masses of two arbitrary and independent knowledge sources m_1 and m_2 into a third BPM m_3 is as follows:

$$m_3\langle Z \rangle = \frac{\Sigma m_1 \langle X_i \rangle * m_2 \langle Y_j \rangle}{1 - k} \tag{2.14}$$

where $Z = X_i \cap Y_j$ and

$$k = \Sigma m_1 \langle X_i \rangle * m_2 \langle Y_j \rangle \tag{2.15}$$

where $X_i \cap Y_j = \varnothing$. Here X_i and Y_j represent various propositions which consist of fault hypotheses and disjunctions thereof. Thus, the BPM of the intersection of X_i and Y_j is the pro-

duct of the individual BPMs of X_i and Y_j. The factor $(1-k)$ is a normalization constant which prevents the total belief from exceeding unity due to the donation of portions of belief to the empty set.

In order to illustrate this rule, consider the combination of m_1 and m_2 when each contains different evidence concerning the diagnosis of a malfunction in the plasma etching application. Such evidence could result from two different sensor readings. In particular, suppose that the sensors have observed that the flow of of one of the etchant gases into the process chamber is too low. Let the simplified frame of discernment θ = {A, B, C, D, E}, where A,...,E symbolically represent the following equipment faults:

A ≡ Mass Flow Controller Miscalibration
B ≡ Gas Line Leak
C ≡ Routing Valve Malfunction
D ≡ Incorrect Sensor Signal
E ≡ The "No-Fault" Condition

These components are illustrated graphically in the partial schematic of the Lam etcher gas flow system shown in Figure 2.3. (For a detailed discussion of the operation of the etcher gas flow system, refer to Chapter 3). Suppose that belief in this frame of discernment is distributed according to the BPMDs m_1 <A $\cup$ B, C, D, E, θ> = <0.48, 0.12, 0, 0.2, 0.2> and m_2 <B, A $\cup$ C, D $\cup$ E, θ> = <0, 0.7, 0.1, 0.2>.

The calculation of the combined BPMD m_3 is visualized in Table 2.1. Each cell of the table contains the intersection of the corresponding propositions from m_1 and m_2 along with the product of their individual beliefs. Note that the intersection of any proposition with θ is the original proposition. The BPM attributed to the empty set, which originates from the presence of various propositions in m_1 and m_2 whose intersection is empty, is:

$$k = (0.48)(0.1) + 0 + (0.12)(0.1) + 0 + 0 + 0 + (0.2)(0.7) = 0.2 \qquad (2.16)$$

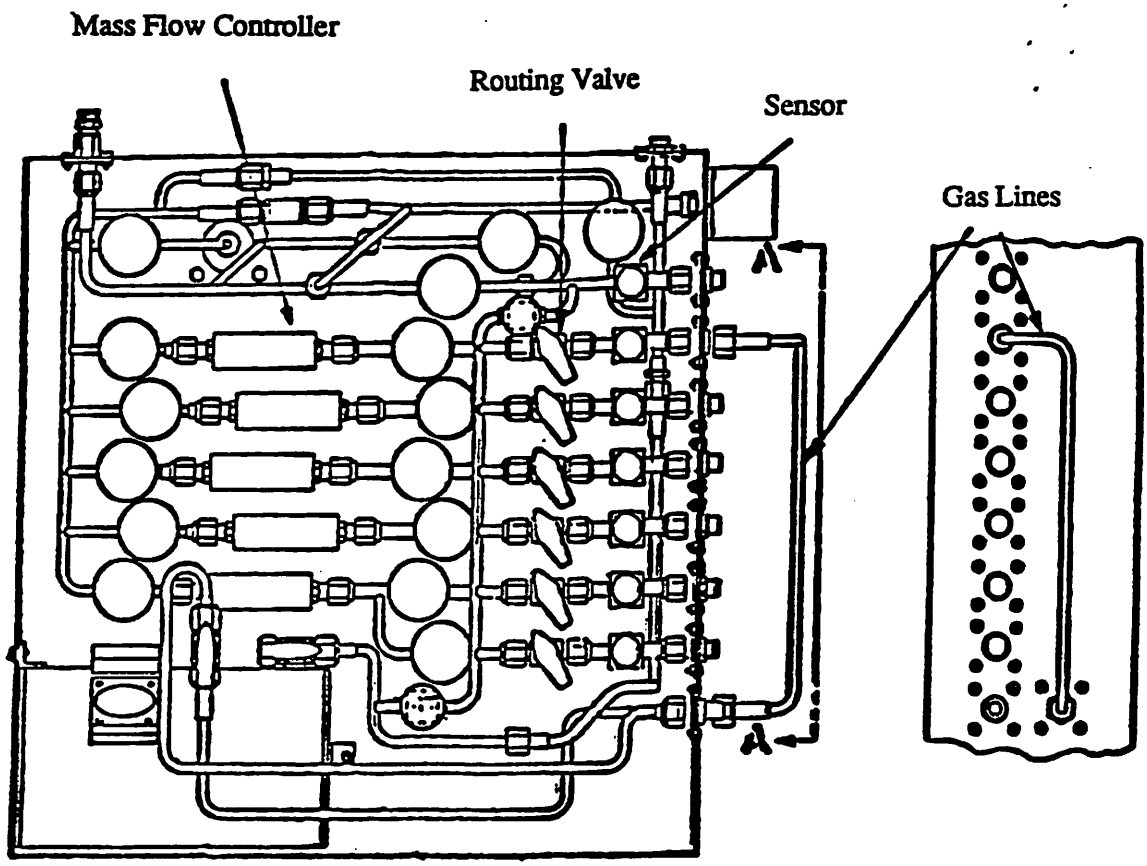


Figure 2.3 - Schematic diagram of the gas system in the Lam plasma etcher [17].

Likewise, the BPM attributed to fault C is m_3<C>, which is given by:

$$m_3<C> = \frac{(0.12)(0.7) + (0.12)(0.2)}{1-k} = 0.135 \tag{2.17}$$

Thus, the products of the individual BPMs in which propositions from m_1 and m_2 intersect donate incremental belief to the combined BPMD. Through the similar application of equation (2.14), the BPMs for the remaining propositions result in the combined BPMD m_3 <A, A $\cup$ B, A $\cup$ C, C, D $\cup$ E, E, θ> = <0.425, 0.12, 0.175, 0.135, 0.025, 0.075, 0.05>.

Table 2.1 - Illustration of BPMD Combination

m_1									
$A \cup B$	0.48	B	0	A	0.34	∅	0.048	$A \cup B$	0.096
C	0.12	∅	0	C	0.084	∅	0.012	C	0.024
D	0	∅	0	∅	0	D	0	D	0
E	0.2	∅	0	∅	0.14	E	0.02	E	0.04
θ	0.2	B	0	$A \cup C$	0.14	$D \cup E$	0.02	θ	0.04
		B	0	$A \cup C$	0.7	$D \cup E$	0.1	θ	0.2

m_2

The plausibilities for the propositions in the above combined BPMD are calculated by applying equation (2.13). In other words:

$$p(A) = 1 - 0.135 - 0.025 - 0.075 = 0.765 \tag{2.18}$$

Consequently, the individual evidential intervals implied by m_3 are: A[0.425, 0.765], B[0.0, 0.165], C[0.135, 0.355], D[0.0, 0.07], and E[0.075, 0.145]. Therefore, combining the evidence available from knowledge sources m_1 and m_2 leads to the conclusion that the most likely cause of the insufficient gas flow malfunction is a miscalibration of the mass flow controller (A) since this fault hypothesis has the highest belief. The order of likelihood for the remaining faults is: routing valve malfunction (C), gas line leak (B), no faults present (E), and an incorrect sensor signal (D), respectively.

2.5 Diagnosing Multiple Faults Using the Dempster-Shafer Methodology

Recall the constraint violations described by equation (2.2). Let the fault groups which lead to these violations be denoted by H. In other words:

$$C^+ \rightarrow H^+ \tag{2.19}$$

$$C^- \rightarrow H^-$$

Thus the occurrence of fault groups H^+ and H^- causes positive and negative constraint violation, respectively. It is reasonable to assume that no fault group can simultaneously lead to both positive and negative deviation. If an existing fault group which is to be diagnosed is denoted by H_i, then the elements of H_i can be enumerated as follows:

$$H_i = \left\{ h_{i1}, h_{i2}, \cdots \right\} \tag{2.20}$$

where h_{ij} are the individual faults within a fault group.

Often, it is assumed that each h_{ij} is a single fault. As described in [13] and [17], this assumption reduces the number of mappings from the evidence space to the fault hypothesis space from 2^θ to θ. This is usually not too restrictive since multiple equipment malfunctions rarely occur independently. Under such an assumption, diagnosis becomes simply a matter of isolating the particular fault responsible for constraint violation by finding the intersection of all fault groups H_i which exist in the system.

However, as has already been pointed out, it is occasionally possible for secondary faults to be triggered by the occurrence of primary faults. In cases such as these, if we relax the single fault assumption and allow individual h_{ij}'s to represent either single faults or simultaneously occurring multiple faults, then the same diagnosis procedure remains equally capable of deducing these multiple faults. Thus, the requirement of conditional independence of evidence that inhibits inference in probabilistic and Bayesian reasoning systems (see § 2.3.2 and 2.3.3) may be relaxed under the Dempster-Shafer framework. To illustrate this point, consider the frame of discernment described in the previous section. Suppose that routing valve malfunctions can cause gas lines to break and subsequently leak. In these cases, it would be useful if malfunction diagnosis could capture the concurrent incidence of both faults. To do so, it would seem that the frame of dis-

cemment should also include an entry for B $\cap$ C. However, this would confuse the diagnosis of cases where either B or C occurred individually.

Nevertheless, it is perfectly valid to represent the intersection of B and C by a new variable, say F. In other words, let F = B $\cap$ C. Then, in diagnosing the malfunction F is treated as just another single fault. Thus, if all combinations of similarly induced multiple faults are known and enumerated in the knowledge base of the diagnostic system, then they may be deduced. In this way, the requirement of independence for different sources of evidence may be circumvented by appropriately grouping faults which may induce other faults. However, it should be noted that this approach is still somewhat limited since a thorough and comprehensive enumeration is a non-trivial task, and an incomplete list means that the accurate diagnosis of multiple faults cannot be guaranteed in all cases.

2.6 Summary

This chapter presents a general method for inferring the malfunctions of semiconductor manufacturing equipment. This method is based upon the application of Dempster-Shafer evidential reasoning techniques. This approach offers several distinct advantages over other inference strategies. It provides a systematic and efficient way to integrate evidence obtained at irregular intervals from multiple knowledge sources in an uncertain measurement environment and use this combined evidence to diagnose a ranked list of possible equipment faults. Moreover, the Dempster-Shafer concept of evidential plausibility is conceptually similar to the statistical notion of *significance probability* [29]. As will be shown later in Chapter 5, this fact makes the Dempster-Shafer approach very attractive for diagnosis based on statistical models [30].

In order to implement the Dempster-Shafer approach on a practical piece of manufacturing equipment, three major bodies of information are required. First, a thorough understanding of the operation of the particular piece of equipment to which the methodology is to be applied is

essential. Such an understanding provides the basis of the Dempster-Shafer frame of discernment which serves as the knowledge base of the system. Equally necessary is a set of governing equations which model the observable equipment behavior. Finally, a technique which generates and distributes numerical evidential support among the various faults in the frame of discernment is the sole remaining portion of a practical diagnostic system that is required to implement the Dempster-Shafer approach. Using the plasma etcher as an application vehicle, the development of these three components of the overall system is the subject of the next three chapters of this thesis.

References for Chapter 2

[1] G. S. May, J. Huang, and C. J. Spanos, "Statistical Experimental Design in Plasma Etch Modeling," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 2, May, 1991.

[2] D. C. Montgomery, *Introduction to Statistical Quality Control*, 2nd Edition, New York: Wiley, 1989.

[3] C. J. Spanos, "HIPPOCRATES: A Methodology for IC Process Diagnosis," *Proceedings of the International Conference on Computer-Aided Design*, 1986.

[4] G. Freeman, J. Y.-C. Pan, and W. Lukaszek, "Application of Analytic Device Models to Automated IC Process Diagnosis," *Proceedings of the Fifth Annual SRC/DARPA CIM-IC Workshop*, August, 1990.

[5] R. Isermann, "Process Fault Detection Based on Modeling and Estimation Methods - A Survey," *Automatica*, vol. 20, no. 4, 1984.

[6] D. M. Manos and D. L. Flamm, *Plasma Etching: An Introduction*, San Diego: Academic Press, 1989.

[7] G. S. May, J. Huang, and C. J. Spanos, "Modeling the Etch Characteristics of Polysilicon in CCl_4 Plasmas Using Response Surface Methodology," *Proceedings of the Ninth IEEE International Electronics Manufacturing Technology Symposium*, 1990.

[8] J. K. Kabarian, "Statistical Diagnosis of IC Process Faults," *Carnegie-Mellon Research Report No. CMUCAD-90-52*, December, 1990.

[9] H. H. Harman, *Modern Factor Analysis*, Chicago: University of Chicago Press, 1967.

[10] J. Y. Pan and J. M. Tenenbaum, "PIES: An Engineer's 'Do-It-Yourself' Knowledge System for Interpretation of Parametric Test Data," *Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 836-843, 1986.

[11] S. B. Dolins, A. Srivastava, and B. E. Flinchbaugh, "Monitoring and Diagnosis of Plasma Etch Processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, no. 1, February, 1988.

[12] K. Parsaye and M. Chignell, *Expert Systems for Experts*, New York: Wiley, 1988.

[13] G. Shafer, "A Mathematical Theory of Evidence," *Princeton University Press*, 1976.

[14] J. F. Baldwin, "Support Logic Programming," *Fuzzy Sets Theory and Applications*, Great Britain: D. Reidel Publishing Company, 1986.

[15] K. Weiskamp and B. Flaming, *The Complete C++ Primer*, San Diego: Academic Press, 1990.

[16] G. S. May and C. J. Spanos, "Automated Malfunction Diagnosis of a Plasma Etcher," *Proceedings of the 1991 International Semiconductor Manufacturing Science Symposium*, May, 1991.

[17] N. H. Chang and C. J. Spanos, "Chronological Equipment Diagnosis with Evidence Integration: An LPCVD Application," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 1, February, 1991.

[18] M. A. Kramer, "Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems," *Journal of the American Institute of Chemical Engineers*, vol. 33, no. 1, January, 1987.

[19] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, vol. 1, March, 1985.

[20] D. S. Vaughan, B. M. Perrin, R. M. Yadrick, and P. D. Holden, "Comparing Expert Systems Built Using Different Uncertain Inference Systems," *Proceedings of the Fifth Annual AAAI Workshop on Uncertainty and AI*, August, 1989.

[21] D. E. Heckerman and E. J. Horvitz, "On the Expressiveness of Rule-Based Systems for Reasoning with Uncertainty," *Proceedings of the Sixth National AAAI Conference on Artificial Intelligence*, July, 1987.

[22] D. E. Heckerman, "Probabilistic Interpretations for MYCIN's Certainty Factors," *Uncertainty in Artificial Intelligence*, L. Kanal and J. Lemmer (editors), North Holland, 1986.

[23] K.-C. Ng and B. Abramson, "Uncertainty Management in Expert Systems," *IEEE Expert*, vol. 5, no. 2, April 1990.

[24] A. Rege and A. M. Agogino, "Topological Framework for Representing and Solving Probabilistic Inference Problems in Expert Systems," *IEEE Transactions on Systems, Man & Cybernetics*, vol. 18, no. 3, 1988.

[25] L. A. Zadeh, "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *Approximate Reasoning in Expert Systems*, M. M. Gupta, A. Kandel, J. B. Kiszka (editors), Elsevier Science Publishers: North Holland, 1985.

[26] L. A. Zadeh, "A Computational Approach to Fuzzy Quantifiers in Natural Languages," *Computation & Mathematics with Applications*, vol. 9, no. 1, 1983.

[27] R. F. Bordley, "Deducing Dempster's Law of Conditioning from a Bayesian Probability Model," Conference Proceedings: *TIMS/ORSA Conference*, New York, 1989.

[28] L. A. Zadeh, "A Simple View of the Dempster-Shafer Theory of Evidence and its Implication for the Rule of Combination," *The AI Magazine*, Summer, 1986.

[29] G. E. P. Box, W. B. Hunter, and J. S. Hunter, *Statistics for Experimenters*, New York: Wiley, 1978.

[30] C. J. Spanos and G. S. May, "Using Regression Equations for Model-Based Diagnosis in the Berkeley Computer-Aided Manufacturing System," *Workshop on Intelligent Diagnostic and Control Systems for Manufacturing*, Boston, July, 1990.

CHAPTER 3

AN OPERATIONAL DESCRIPTION OF THE LAM RESEARCH AUTOECH 490 PLASMA ETCHER

3.1 Introduction

Wet etching was the standard method of pattern transfer in early generations of integrated circuits. This stemmed primarily from the fact that etchants with high selectivity to both the substrate and the masking layer were readily available. However, wet etching processes are almost invariably isotropic in nature. Consequently, when the thickness of the film being etched becomes comparable to the minimum pattern dimension (as is the case with modern VLSI circuits), the undesirable lateral undercut due to the etch isotropy of wet etchants is no longer tolerable. In order to overcome the shortcomings of wet etch processes, the technique of ion-assisted plasma etching has become widely used in semiconductor manufacturing [1]. Since this method offers the added feature of etch anisotropy, considerable effort has been expended in recent years to develop plasma etch processes.

As is the case with other semiconductor manufacturing equipment, faulty or unreliable etcher performance can jeopardize the quality of integrated circuits. Moreover, since the duration of plasma processes is typically on the order of minutes, very rapid malfunction diagnosis in plasma etchers is necessary in order to minimize misprocessing. This is especially true during the fabrication of MOS circuits, where plasma etching is essential to the definition of polysilicon gates [2]. However, etch diagnosis is further complicated by the fact that plasma processes are currently not well understood [3]. Since the problem of etch diagnosis is lacking in a solid physical foundation for reasoning, the plasma etcher makes a very suitable candidate for the application of the approach to equipment diagnosis described in the previous chapter.

One of the most basic and essential components of any expert system is its *knowledge base* [4-5]. In this respect, diagnostic expert systems are no exception. The knowledge base represents the compilation of the various facts which the system uses to perform its reasoning functions. In the case of the proposed diagnostic system for the plasma etcher, the knowledge base is a formalized description of the etcher and its operation. It consists of a catalog of various equipment faults as well as information concerning possible causes of those faults. This catalog defines the diagnostic *frame of discernment* [6].

Since each malfunction which the expert system attempts to diagnose results from a particular faulty component, the development of the knowledge base for a system designed to troubleshoot equipment failures requires a relatively thorough understanding of the functionality of that particular piece of equipment at a component level [7]. Consequently, this chapter gives an overview of the operation of the Lam Research Autoetch 490 Plasma Etching System as well as a description of the manner in which various etcher components are categorized as diagnosable faults.

3.2 System Operation Overview

The Lam Autoetch 490 is a cassette-to-cassette, fully automated, single-wafer parallel-plate system. Automated functions are controlled by means of a Z80 microprocessor. Etching programs may be entered manually from the keyboard and saved on a recipe programming module. Programmable recipe parameters include operating pressure, RF power, electrode spacing, and gas flow rates.

The entire Autoetch system is under automatic, closed-loop control and is double key-locked to prevent accidental alteration of the process. In addition, the Autoetch has a continuous CRT display for monitoring the machine status and the process parameters. This system is designed to monitor and report on its own performance through the communications interface

modem to a remote diagnostics station [8].

The entire machine (schematic shown in Figure 3.1) consists of several basic subsystems, including the RF power system, electronics, wafer transport, temperature control, the process chamber, the vacuum system, and the gas system [9]. Each subsystem is described in detail below. Component failures are catalogued in a hierarchical fashion according to the Lam subsystem to which each component belongs. This hierarchy, which is presented in Appendix 3.1, is essentially a simplified overview of common Autoetch malfunctions obtained through extensive consultation with the equipment maintenance technician serving as the domain expert [10]. This equipment description facilitates the inclusion of the Lam etcher in FAULTS, the BCAM computerized maintenance record-keeping system [11].

3.2.1 RF Power System

The Autoetch is equipped with solid state RF power supplies which are water-cooled, 650 watt or 1.25 kilowatt generators. These generators are automatically controlled and operate at 13.56 MHz with an output impedance of 50 Ω. The generators are completely enclosed to insure operator safety.

As process parameters change, the electrical impedance of the etch process chamber also changes. Therefore, the RF power is connected to a match assembly in which the chamber impedance is matched to the generator output impedance by means of a phase/magnitude detector system. Matching is necessary to achieve maximum power transfer. The RF matching network automatically matches the impedance of the plasma to that of the RF generators as power, electrode spacing, gas composition, and chamber pressure vary.

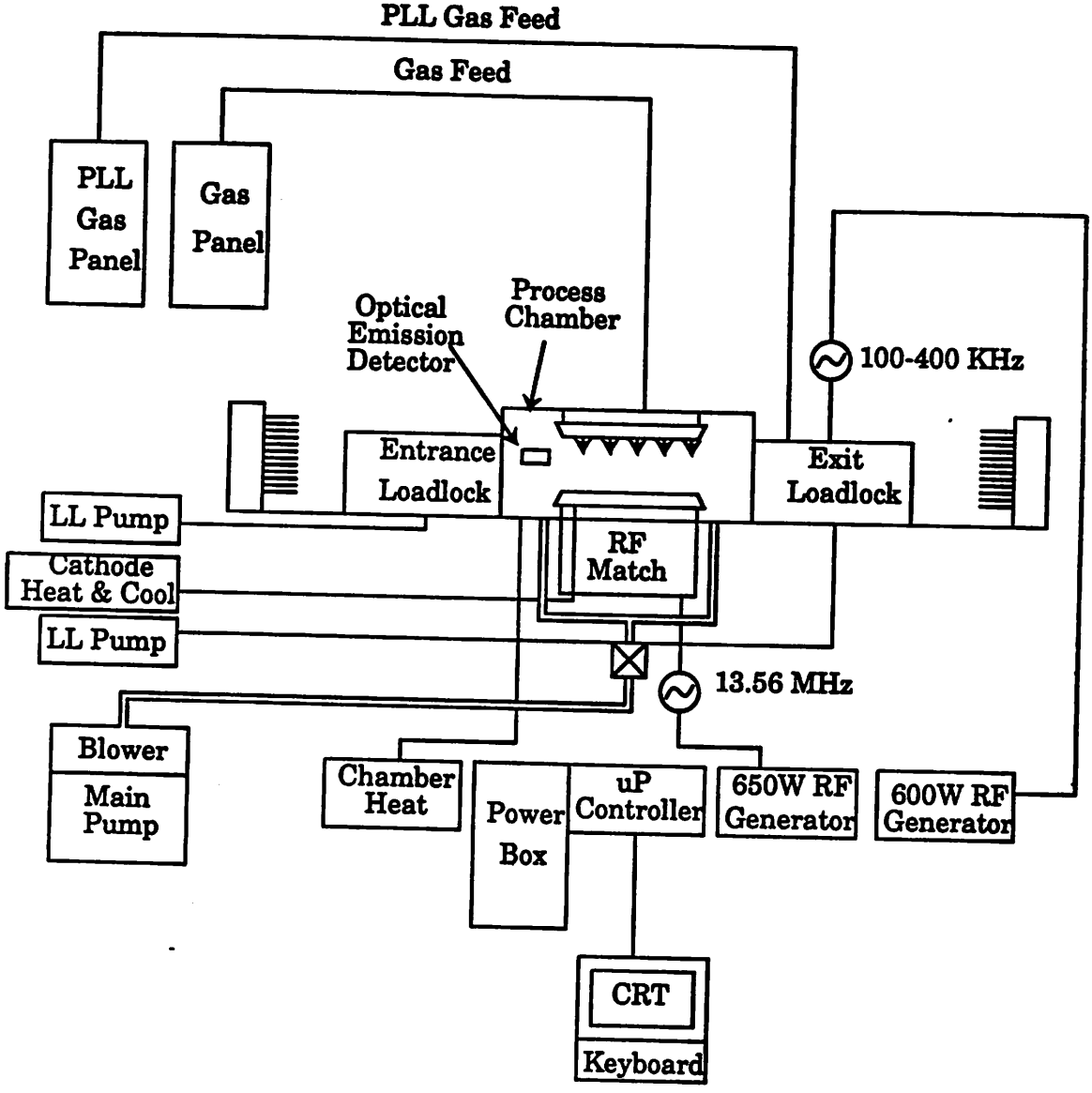


Figure 3.1 - Schematic diagram of the Lam Research Autoetch 490.

3.2.2 System Electronics

Etcher electronics consist of a controller drawer assembly, a power distribution box assembly, and interface electronics. The controller drawer houses a microprocessor card cage, various printed circuit boards, and power supplies. A microprocessor controls all the functions of the

machine and memory cards contain all equipment instructions. TTL driver cards operate solenoids, relays, and motors. Analog and digital input boards provide information to the microprocessor CPU concerning machine status.

The power distribution box assembly distributes the required AC and DC voltages necessary for the system to function properly. The electronics system contains among other things, a resistor thermometer device (see § 3.2.4) interface board, a guard seal interface board, and RF generator interface board, an endpoint detector board, and a modem which allows communication between the Autoetch and a remote monitoring system.

3.2.3 Wafer Transport System

Wafers are loaded from cassettes into and out of the process chamber by means of Hine indexers mounted on the upper front plates of the Autoetch and pneumatic station lifters. Send indexers load individual wafers onto the motor-driven wafer belt conveyer system. Receive indexers take finished wafers from the belt drive system and store them in the receive cassette.

The entrance station lifter elevates unprocessed wafers to a point where they may be reached by the entrance airlock arm. Similarly, the exit station lifter receives finished wafers from the exit airlock arm and lowers them to the receiver conveyer belt. Lifters are raised and lowered by pneumatically driven pistons. Wafers are held onto the lifters by a vacuum generated by the airlock vacuum manifold which is controlled by a solenoid valve actuated by the microprocessor CPU (see § 3.2.2).

An infrared sensor on the entrance station informs the etcher that a wafer is present. There is also an additional sensor mounted ahead of the entrance station which senses the presence of more than one wafer. If multiple wafers are detected, the lifter will not rise and the process is halted.

The etcher airlocks act as a buffer between the clean room environment and the process

chamber since wafers are loaded and unloaded from the chamber through them. Closing the inner doors isolates the airlocks from the chamber and allows the airlocks to be vented. Thus, these airlocks allow the chamber to remain in a constant state of vacuum. The inner and outer airlock doors are operated by two air cylinders, a gear train, and a linkage mechanism which is designed to insure that the door is mechanically locked while wafers are being processed.

Airlock arms are driven by two air cylinders in series. The translation of the linear motion of these cylinders to the motion that is required to extend and retract the arm of the airlock is achieved through another mechanical gear train. Mechanical interlocks assure that the airlock doors cannot be closed until the arm is centered in the airlock.

3.2.4 Temperature Control System

The lower electrode of the Autoetch process chamber is cooled by a temperature control system which removes heat from the electrode by circulating deionized water. This closed-loop, recirculating system is know as the "chiller". The chiller maintains a preset temperature level. For the etcher in the Berkeley microlab, this level is set to room temperature. The temperature is controlled by means of a resistance thermometer device (RTD) located on the bottom of the electrode which sends temperature information to the CPU. The etcher also has a low water flow alarm switch which monitors the level of the deionized water from the chiller.

3.2.5 Process Chamber

It is inside the process chamber that wafers are etched by computer-controlled gas discharge. Therefore, the process chamber must remain under vacuum at all times. The chamber is located directly behind the operator interface display. This display is hinged and will swing upward to allow direct visual access to the process chamber. The chamber is equipped with quartz windows on the front and rear. Each window has a wire mesh screen to contain the RF

field and a plexiglass cover for UV filtering. Not only do these windows allow for observation of the etch process, but the front window plate also serves as the mounting point for the capacitance manometer (see § 3.2.6).

The major subsystems of the process chamber are the upper and lower electrode assemblies, the electrode gap adjustment system, and the RF match assembly (see § 3.2.1). The lower electrode assembly contains the anode on which the wafer rests during processing. The gap adjustment system provides for modification of the space between the cathode (upper electrode) and the wafer. The gap housing has two large guard seals which provide a vacuum seal between the chamber and atmospheric pressure.

3.2.6 Chamber/Airlock Vacuum System

The airlock vacuum system can be divided into the airlock vacuum pump assembly and the airlock manifold. The vacuum pump is installed remotely from the etcher itself. It consists of an Edwards EH 250 Roots Blower and an E2M40F Rotary Pump. Control of the chamber pressure is achieved through a "butterfly" throttle valve attached to the vacuum manifold at the rear of the machine. An automatic throttle valve controller maintains chamber pressure by comparing the pressure setpoint given in the etch recipe with the actual chamber pressure as monitored by a capacitance manometer. The chamber vacuum manifold, with a pneumatically operated isolation valve, is attached to the chamber vacuum foreline assembly. The isolation valve is normally closed and is activated by the CPU during operation of the system.

3.2.7 Gas System

The gas panel assembly independently regulates the flow of up to five gases into the process chamber. These gases flow into the chamber according to the flow rates indicated by the etch recipe to create the various plasmas. Gases are connected to the rear panel of the machine with

aluminum fittings and each passes through a filter before entering a mass flow controller (MFC). While not being actively controlled, the MFC's go into a "normally open" state and then close to regulate the flow of gas. There is a pneumatically operated valve both upstream and downstream from each MFC. There are also two valves which rout the gases either to the process chamber or into the chamber vacuum manifold.

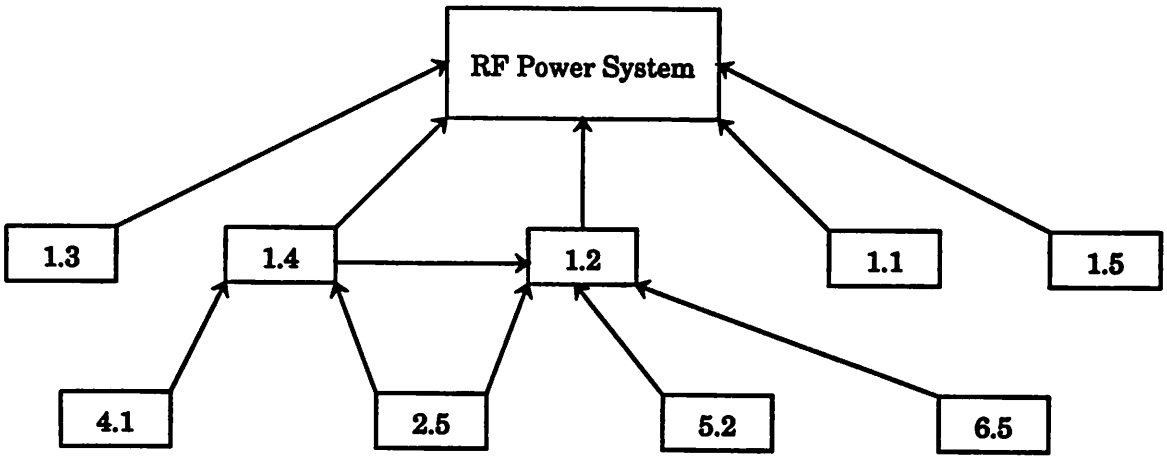
On polysilicon etch machines, the gas panel also houses a CCl_4 tank. This tank holds the CCL_4 or other process liquids under vacuum and stores the gas until it is required for etching. The desired flow of gas is programmed from the recipe and causes an analog signal to be sent to each MFC. The necessary amount of gas will then flow through the gas panel MFC's into the process chamber. The line between the gas panel and the process chamber is a quarter-inch stainless steel flexible tube housed in a corrugated tube.

3.3 Fault Propagation

In complex electromechanical systems such as a plasma etcher, equipment malfunctions are frequently propagated from one subsystem to another. For example, it is conceivable that a deionized water flow sensor miscalibration could cause insufficient coolant to circulate through the chiller. This in turn can cause RF power supplies to overheat. The ultimate result of such a problem would cause RF power mismatch or instability errors. The only observable symptom in this case indicates a fault in the RF matching network. However, the actual root cause of the malfunction was the incorrect water flow sensor.

In order to insure accurate and robust diagnosis, it is therefore necessary to obtain a comprehensive mapping of all potential fault propagation patterns. For the Lam Autoetch 490, these relationships have been captured by means of lengthy discussions with maintenance technicians [10] as well as thorough inspection of the recorded equipment maintenance history. This information is visually summarized in "System Fault Propagation Diagrams," such as the one that

appears in Figure 3.2. These diagrams are very similar to *influence diagrams*, which are often used in expert systems to graphically represent relationships critical to the problem under consideration [12]. This particular figure shows fault propagation in the Autoetch RF Power System. It includes the flow sensor malfunction scenario descibed above as well as other potential problems related to RF power.



1.1 Interconnect
1.2 RF Matching Network
1.3 Phase/Magnitude Detector
1.4 Power Supplies
1.5 Generator Calibration
2.5 Interface Electronics
4.1 Chiller
5.2 Electrode Gap Adjustment System
6.5 Pressure Controller "Butterfly" Valve

Figure 3.2 - Fault propagation diagram for Lam Autoetch 490 RF power subsystem. Arrows indicate the direction which component malfunctions may influence subsequent failures. (Component numbering refers to the classification system given in Appendix 3.1).

In general, these diagrams are intended to depict the interrelationships of all subsystems and components in the etcher. These relationships are critical to the development of a useful diagnostic tool since they describe the manner in which faults evolve from root causes to observable

symptoms. The complete set of fault propagation diagrams for the Autoetch 490 appears in Appendix 3.2.

3.4 Summary

This chapter has provided a description of the operation of the Lam Research Autoetch 490. Since equipment malfunctions result from faulty components, this information is essential to the development of the knowledge base of the system intended to diagnose such malfunctions in the etcher. Another necessary component of the diagnostic system is a system of models to describe equipment behavior. The derivation of such models requires a thorough characterization of the response of process outputs to variations in input parameters. Model development through this process characterization is the subject of the next chapter.

References for Chapter 3

[1] D. M. Manos and D. L. Flamm, *Plasma Etching: An Introduction*, San Diego: Academic Press, 1989.

[2] D. H. Bower, "Planar Plasma Etching of Polysilicon Using CCl_4 and NF_3," *Journal of the Electrochemical Society*, vol. 129, no. 4, April, 1982.

[3] G. S. May, J. Huang, and C. J. Spanos, "Statistical Experimental Design in Plasma Etch Modeling," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 2, May, 1991.

[4] K. Parsaye and M. Chignell, *Expert Systems for Experts*, New York: Wiley, 1988.

[5] P. Wallich, "Software 'Doctor' Prescribes Remedies," *IEEE Spectrum*, October, 1986.

[6] M. A. Kramer, "Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems," *Journal of the American Institute of Chemical Engineers*, vol. 33, no. 1, January, 1987.

[7] A. M. Agogino, R. K. Paasch, K. A. Swanson, R. E. Heiskell, J. M. Quinto, and R. G. Taylor, "CVAID: An Expert System for Diagnosis and Repair of a Car Valve Actuator," *Computers in Engineering 1988*, vol. 1, 1988.

[8] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, March, 1985.

[9] "Advanced Dry Etching of Aluminum and its Alloys," *Solid State Technology*, April, 1986.

[10] R. Norman, *Private Communication*, November, 1989.

[11] D. Mudie and N. H. Chang, "FAULTS: An Equipment Maintenance and Repair System Using a Relational Database," *Proceedings of the 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium*, October, 1990.

[12] A. Rege and A. M. Agogino, "Topological Framework for Representing and Solving Probabilistic Inference Problems in Expert Systems," *IEEE Transactions on Systems, Man & Cybernetics*, vol. 18, no. 3, July, 1988.

APPENDIX 3.1

LAM ETCHER FAULT DESCRIPTION

Below is a hierarchical description of the major systems, subsystems and components of the Lam Research Autoetch 490 Plasma Etch System. Each item in this listing is accompanied by an abbreviated designation which is used in FAULTS, the BCAM automated equipment maintenance record-keeping system.

1. RF POWER SYSTEM (rf)

1.1 Interconnect (cable-problem)

1.2 RF Matching Network (matching-network)

1.3 Phase/Magnitude Detector (phase-mag-detect)

1.4 Power supplies (power-supply)

1.5 Generator Calibration (rf-calibration)

2. SYSTEM ELECTRONICS (computer)

2.1 24-volt DC Power Supply (24volt-supply)

2.2 Circuit Boards (board-failure)

2.3 Endpointer (endpointer)

2.4 Equipment Communication (host-link)

2.5 Interface Electronics (interface)

2.6 Power Supply (power-supply)

2.7 Recipe Module (recipe-module)

3. WAFER TRANSPORT SYSTEM (wafer-transport)

3.1 Airlock Arm (arm-adjustment)

3.2 Belts (belt)

3.3 Belt Drive System (belt-motor)

3.4 Airlock Doors (door-adjustment)

3.5 Hine Indexer (cassette-indexer)

3.6 Door O-Rings (o-rings)

3.7 Transport Pneumatics (pneumatics)

3.8 Wafer Presence Sensors (sensor-adjustment)

4. TEMPERATURE CONTROL SYSTEM (cooling)

4.1 Chiller (chiller-problem)

4.2 Chiller Water Level (chiller-water-level)

4.3 Water Flow Sensor (flow-sensor)

4.4 Water Utility (recirc-water)

4.5 Procon Pump (mechanical-pump)

5. PROCESS CHAMBER (chamber)

5.1 Endpoint Window (endpoint-window)

5.2 Electrode Gap Adjustment System (gap)

5.3 Guard Seals (guard-seals)

5.4 Leaks (leak)

5.5 Lower Electrode Assembly (lower-assembly)

5.6 Polymer Deposition (polymer-deposition)

5.7 Wafer in System (wafer-in-system)

5.8 Worn Electrodes (worn-electrodes)

6. CHAMBER/AIRLOCK VACUUM SYSTEM (vacuum)

6.1 Roots Blower (blower)

6.2 Capacitance Manometer (cap-manometer)

6.3 Isolation Valves (isovalves)

6.4 Chamber Vacuum Manifold (manifold-leak)

6.5 Pressure Controller "Butterfly" Valve (press-control)

7. GAS SYSTEM (gas)

7.1 CCl_4 Tank (ccl4-fill)

7.2 Gas Cylinders (gas-cylinder-empty)

7.3 Gas Lines (gas-lines)

7.4 MFC Calibration (MFC-cal)

7.5 Routing Valves (valves)

8. RECIPE ERROR (recipe-error)

9. ROUTINE MAINTENANCE (maintenance)

9.1 Analog Interface Calibration (analog-calibration)

9.2 Chamber Clean (chamber-clean)

9.3 Electrode Replacement (electrode-replace)

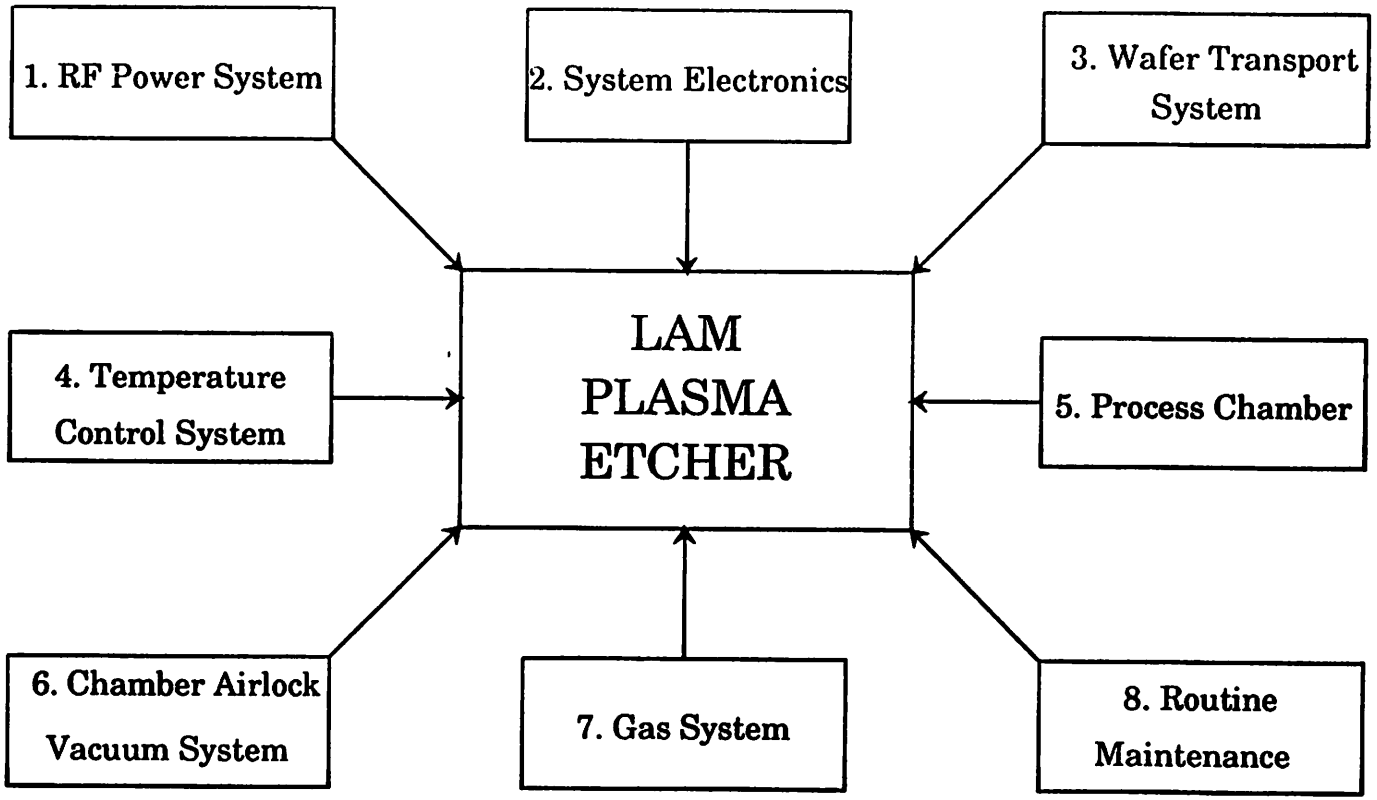
9.4 Change Pump Oil (pump-service)

APPENDIX 3.2

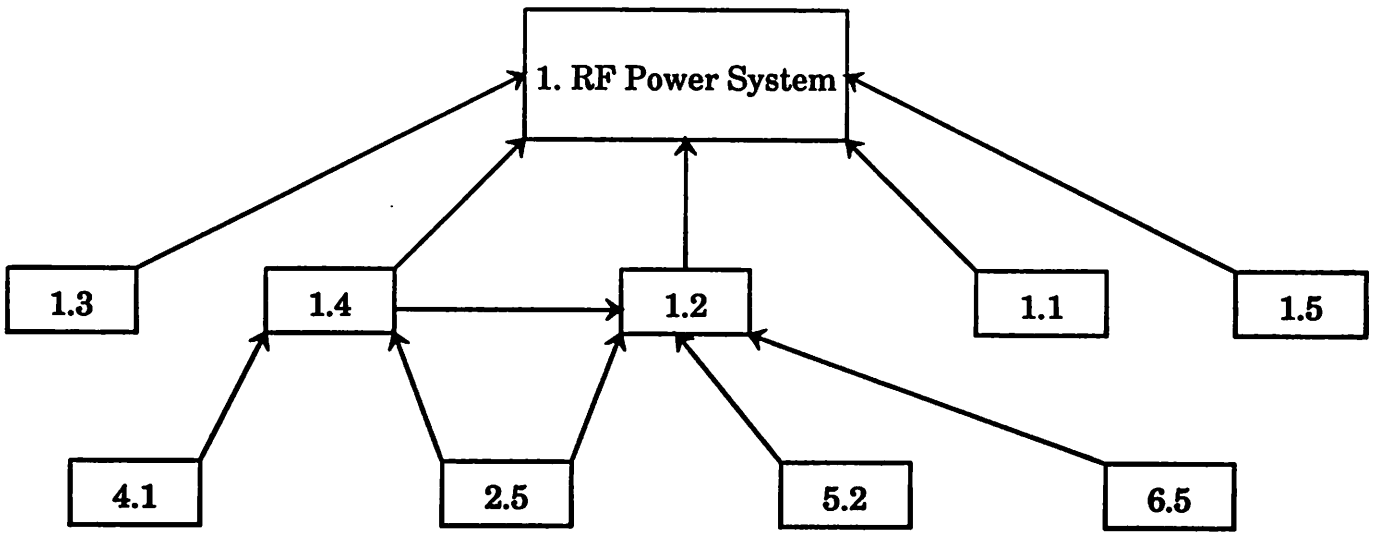
LAM ETCHER FAULT PROPAGATION DIAGRAMS

The following are system fault propagation diagrams for the Lam Research Autoetch 490 Plasma Etch System. The first diagram shows the relationship between each major subsystem and the overall piece of equipment. Subsequent diagrams depict similar relationships and interrelationships for each subsystem at the component level. For each diagram, arrows indicate the direction of malfunction propagation. The component numbering system reflects the classification scheme in Appendix 3.1.

SYSTEM FAULT PROPAGATION DIAGRAM



SYSTEM FAULT PROPAGATION DIAGRAM



1.1 Interconnect

1.2 RF Matching Network

1.3 Phase/Magnitude Detector

1.4 Power Supplies

1.5 Generator Calibration

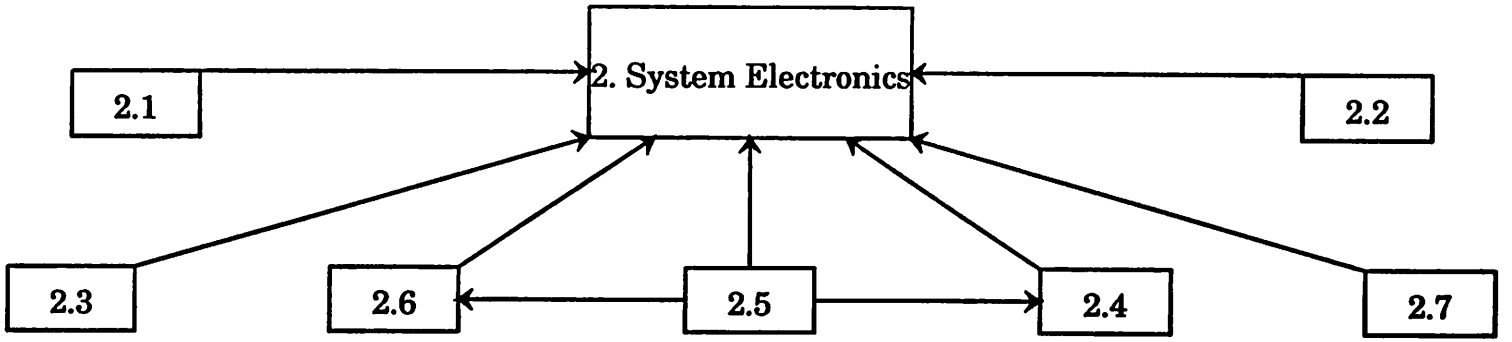
2.5 Interface Electronics

4.1 Chiller

5.2 Electrode Gap Adjustment System

6.5 Pressure Controller "Butterfly" Valve

SYSTEM FAULT PROPAGATION DIAGRAM



2.1 24-volt DC Power Supply

2.2 Circuit Boards

2.3 Endpointer

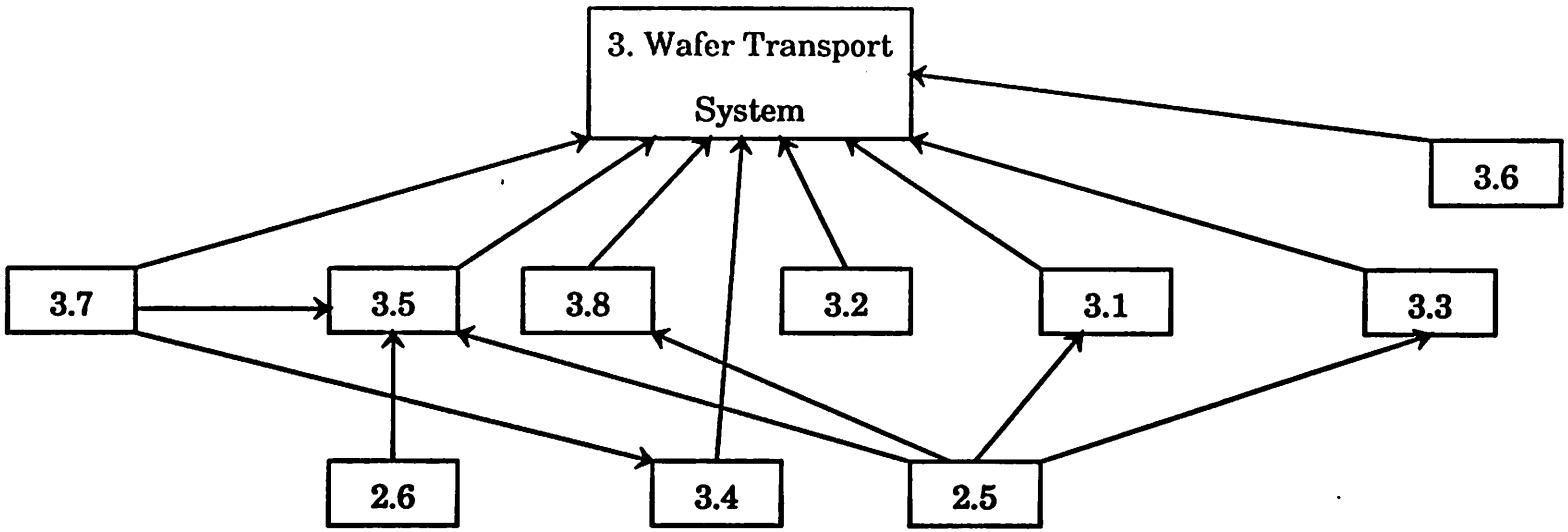
2.4 Equipment Communication

2.5 Interface Electronics

2.6 Power Supply

2.7 Recipe Module

SYSTEM FAULT PROPAGATION DIAGRAM



2.5 Interface Electronics

2.6 Power Supply

3.1 Airlock Arm

3.2 Belts

3.3 Belt Drive System

3.4 Airlock Doors

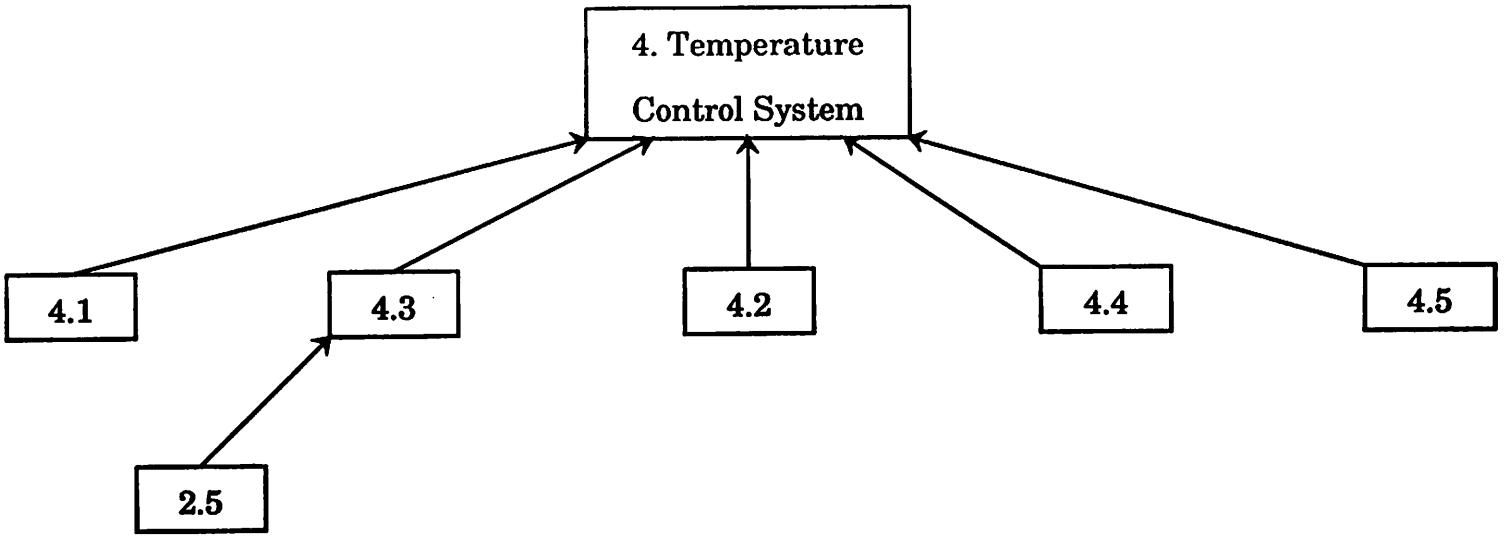
3.5 Hine Indexer

3.6 Door O-Rings

3.7 Transport Pneumatics

3.8 Wafer Presence Sensors

SYSTEM FAULT PROPAGATION DIAGRAM



2.5 Interface Electronics

4.1 Chiller

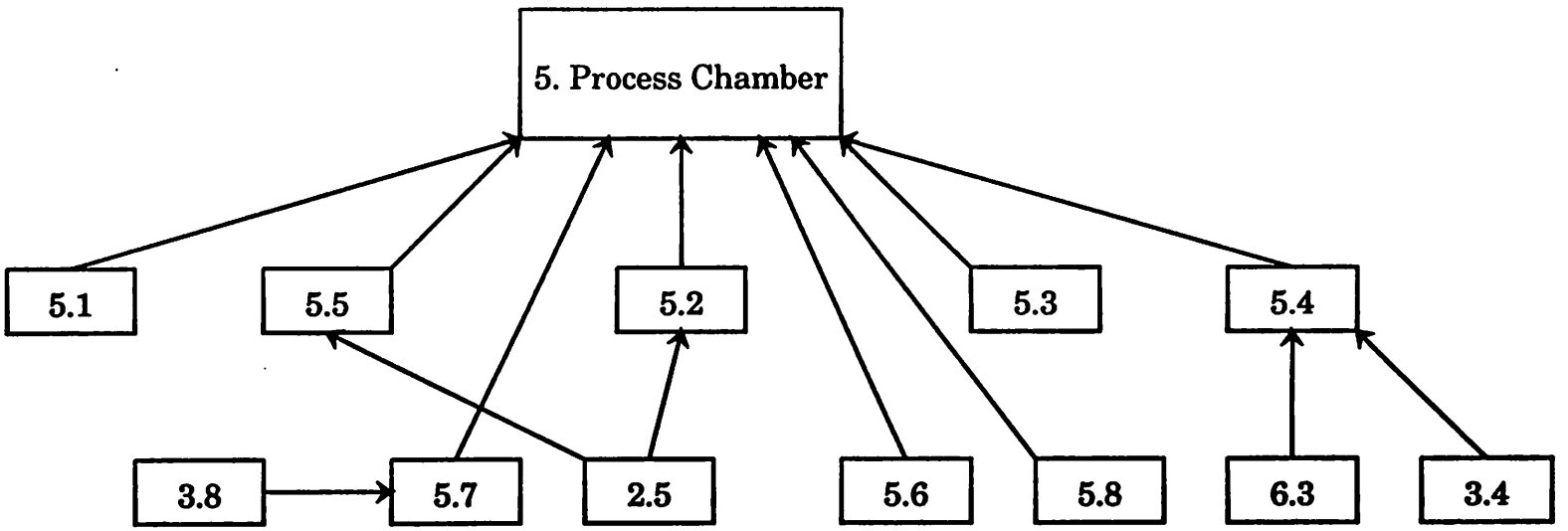
4.2 Chiller Water Level

4.3 Water Flow Sensor

4.4 Water Utility

4.5 Procon Pump

SYSTEM FAULT PROPAGATION DIAGRAM



2.5 Interface Electronics

3.4 Airlock Doors

3.8 Wafer Presence Sensor

5.1 Endpoint Window

5.2 Electrode Gap Adjustment System

5.3 Guard Seals

5.4 Leaks

5.5 Lower Electrode Assembly

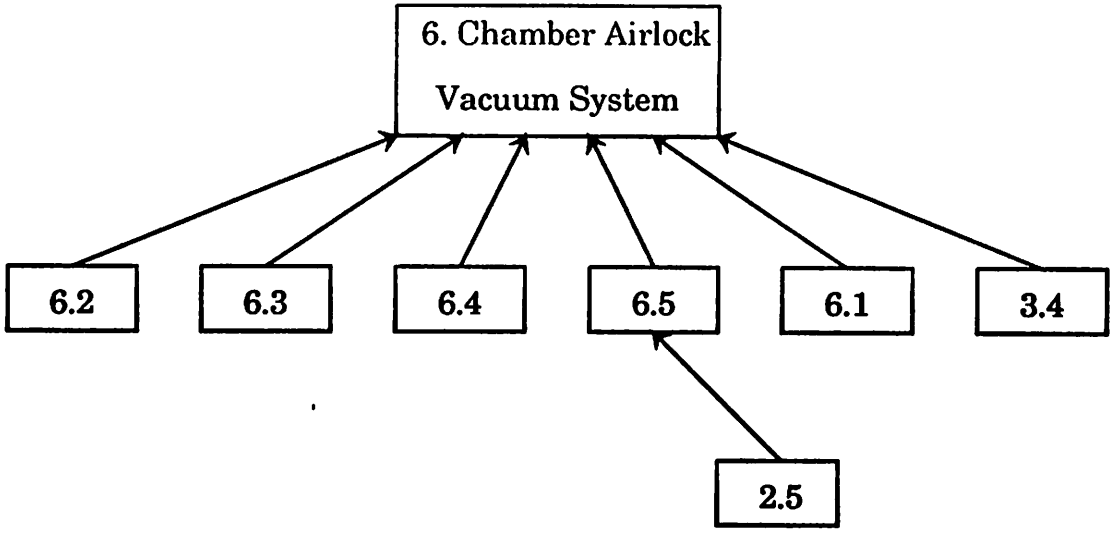
5.6 Polymer Deposition

5.7 Wafer-In-System

5.8 Worn Electrodes

6.3 Isolation Valves

SYSTEM FAULT PROPAGATION DIAGRAM



2.5 Interface Electronics

3.4 Airlock Doors

6.1 Roots Blower

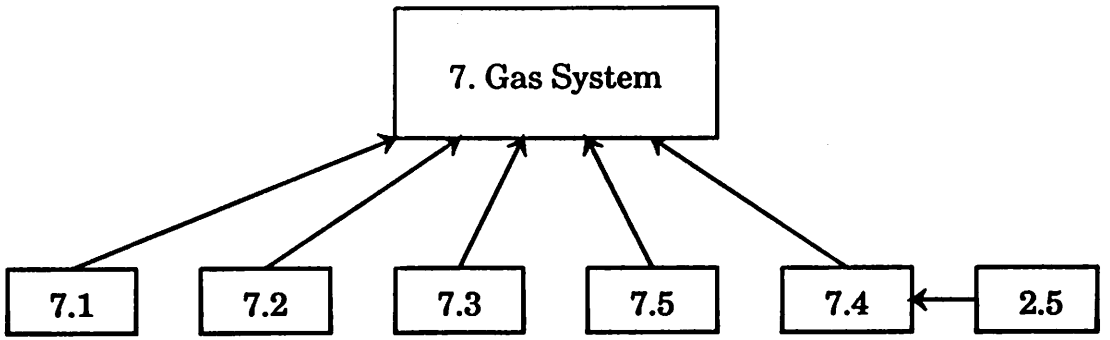
6.2 Capacitance Manometer

6.3 Isolation Valves

6.4 Chamber Vacuum Manifold

6.5 Pressure Controller "Butterfly" Valve

SYSTEM FAULT PROPAGATION DIAGRAM



2.5 Interface Electronics

7.1 CCl-4 Tank

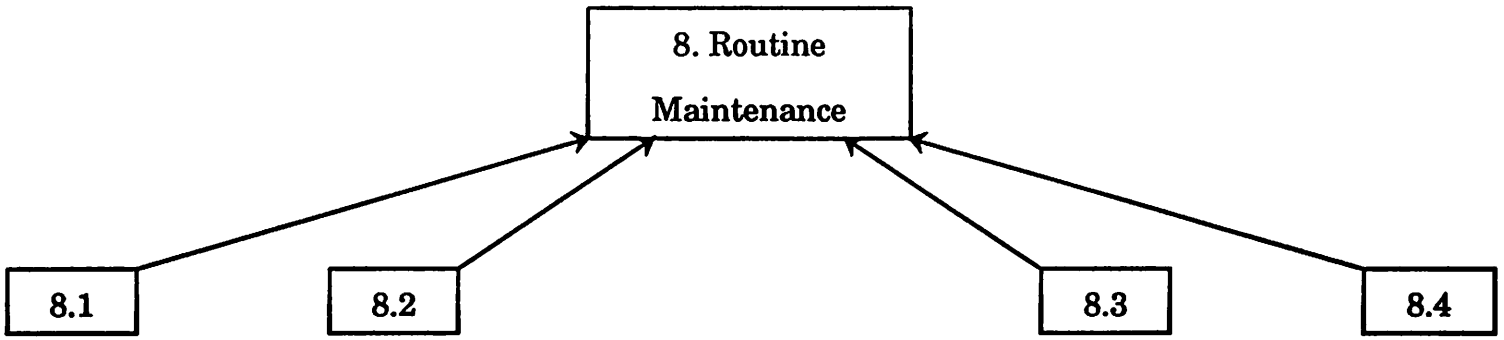
7.2 Gas Cylinders

7.3 Gas Lines

7.4 Mass Flow Controller

7.5 Routing Valves

SYSTEM FAULT PROPAGATION DIAGRAM



8.1 Analog Interface Calibration

8.2 Chamber Clean

8.3 Electrode Replacement

8.4 Change Pump Oil

CHAPTER 4

EXPERIMENTAL MODELING OF THE PLASMA ETCHER

4.1 Introduction

Plasma etch modeling from a fundamental physical standpoint has had limited success. The plasma chemistry of many discharges cannot be modeled because the principal reactions and rate constants are unknown [1]. The best physically-based models currently available are capable of describing the chemical kinetics of one-dimensional RF glow discharges [2-4]. Two-dimensional models of RF glow discharges are now at an early stage of development [5]. These models attempt to derive self-consistent solutions to first-principle equations involving continuity, momentum balance, and energy balance inside a high frequency, high intensity electric field. This is accomplished by means of computationally expensive numerical simulation methods which typically produce outputs such as profiles of the distribution of electrons and ions within the plasma sheath. However, although detailed simulation is useful for equipment design and optimization, it is subject to many simplifying assumptions. Due to the extremely complex nature of particle dynamics within a plasma, the connection between these microscopic models and macroscopic parameters such as etch rate has yet to be clearly distinguished.

Since the complexity of practical plasma processes at the equipment level is presently far ahead of theoretical comprehension, other efforts have focused on empirical approaches to plasma modeling involving Response Surface Methods (RSM) [6-7]. These techniques have been used by several authors to obtain statistical models of the etch rates of various thin films [8-14]. Karulkar and Wirzbicki studied the etch rate of silicon dioxide and positive photoresist in a $CHF_3/C_2F_6/O_2/He$ plasma as a function of oxygen flow, reactor pressure and average RF power [8]. In addition, Riley et. al. investigated the etching of SiO_2 in C_2F_6/CHF_3 plasmas versus flow

rate and pressure [9]. Jenkins et. al. provides a model of the etch rate of p-doped polysilicon in a CF_3Cl/Ar plasma versus pressure, RF power and CF_3Cl fraction [11]. Thompson and Sawin studied the etching of n^+-doped poly in SF_6 plasmas versus power, pressure, and flow rate [12]. Further, Gogolides and Sawin performed a similar characterization of n^+ poly in CCl_4/He plasmas as functions of RF power, pressure and He fraction [13]. Riley and Hanson, on the other hand, investigated silicon nitride etching in SF_6/He versus the combined SF_6/He flow rate, pressure, power and electrode spacing [14].

In many of these studies, however, the complete characterization of other critical process outputs which directly affect product quality such as etch uniformity, selectivity and anisotropy has been somewhat overlooked. Such a complete characterization is a necessary component of any effective diagnostic system. Therefore, the objective of the following experiment was to obtain a comprehensive set of empirical models for plasma etch rates, uniformity, selectivity and anisotropy. These models accurately represent the behavior of a specific piece of equipment under a wide range of etch recipes, thus making them ideal for equipment diagnosis and other manufacturing purposes. In particular, this study focuses on the etch characteristics of n^+-doped polysilicon in a $CCl_4/He/O_2$ plasma. Responses were modeled under the variation of the following six input parameters: RF power, pressure, electrode spacing, and the three gas flows. Etching took place in the Lam Research Autoetch 490 single-wafer plasma system.

4.2 Experimental Design

A prime example of a fabrication step in which plasma etching has become essential is the definition of polysilicon features for MOS circuits. This process step often requires that a relatively thick polysilicon gate be etched down to a thin silicon dioxide layer. Therefore, high selectivity between poly and SiO_2 is necessary in order to use a thin gate oxide as an etch stop. In addition, it is desirable that the vertical etch rate of the polysilicon be much greater than its

horizontal rate to achieve high etch anisotropy. Finally, good within-wafer uniformity and selectivity to photoresist are also desirable. Carbon tetrachloride has been reported as an anisotropic etchant with a high selectivity for polysilicon in plasma etching [15], thus making it an attractive candidate for this experiment.

The most critical control parameters in plasma etching are RF power, chamber pressure, electrode spacing and gas flow [11,14-16]. Helium is often added to standard CCl_4 etch recipes in order to enhance etch uniformity. In addition, oxygen is sometimes also introduced into the gas mixture to suppress polymer deposition in the process chamber [1,9,13]. The effects of all six process variables must be considered in plasma recipe control. However, RSM techniques are most effective when the number of input factors is limited to six or fewer [11,17]. As a result, it was appropriate to divide the overall experiment into an initial *variable screening* [7] phase to determine the most significant parameters, followed by a second phase designed to obtain the statistical response models.

Occasionally, temporary disturbances might affect the operation of a piece of equipment as complex as a plasma etcher. In order to make sure that such disturbances do not occur during the characterization runs, it is important to bring and maintain the operation under Statistical Process Control (SPC). This was accomplished by applying a real-time monitoring system during the experiment. This system was designed with a client-server interface between the etcher and a host computer. A C program allowed the interpretation of Semiconductor Equipment Communications Standard II (SECS-II) messages issued from the etcher showing various sensor values which monitored real-time conditions in the process chamber. The parameters monitored included:

1. CCl_4 flow
2. O_2 flow
3. He flow

4. Forward RF power
5. Reflected RF power
6. Chamber pressure
7. Electrode temperature
8. Electrode gap
9. Guard seal pressure
10. Entrance airlock pressure
11. Exit airlock pressure

Control was ensured by comparing the patterns observed during the experimental runs with similar historical data [18].

4.2.1 First Phase - Screening Experiment

The six factors chosen for the initial screening phase of this experiment along with their respective ranges of interest are shown in Table 4.1. These ranges were chosen to effectively encompass the wide variety of etch recipes currently being utilized in the Berkeley Microfabrication Laboratory. A full factorial experiment to determine all effects and interactions for six factors would require 2^6, or 64 experimental runs. In order to reduce experimental costs, the effects of higher order interactions were neglected and a 2^{6-1} fractional factorial design requiring only 32 runs was performed [19]. This design used a *Resolution V* format which prevented main effects from being confounded with other main effects as well as with two- and three-factor interactions. It also prevented the confounding of two-factor interactions with each other [7].

Table 4.1: Range of Input Factors

Parameter	Range	Units
RF Power	300 - 400	watts
Pressure	200 - 300	mtorr
Electrode Spacing	1.2 - 1.8	cm
CCl_4 Flow	100 - 150	sccm
He Flow	50 - 200	sccm
O_2 Flow	10 - 20	sccm

Table 4.2: Design Matrix for Screening Experiment

Run	Pressure	RF Power	CCl_4 Flow	*He* Flow	O_2 Flow	Electrode Gap	Block
1	300	300	100	200	20	1.8	2
2	200	400	100	50	10	1.8	2
3	200	400	150	200	20	1.2	1
4	300	400	150	200	20	1.8	2
5	200	400	150	50	10	1.2	1
6	300	300	150	200	10	1.8	1
7	300	400	100	50	20	1.8	1
8	250	350	125	125	15	1.5	1
9	200	300	150	200	20	1.8	2
10	300	400	150	50	20	1.2	2
11	300	300	100	200	10	1.2	2
12	200	300	150	200	10	1.2	2
13	200	400	100	200	10	1.2	2
14	300	400	150	50	10	1.8	2
15	200	300	100	50	20	1.8	1
16	200	400	100	200	20	1.8	2
17	200	300	100	200	20	1.2	1
18	300	300	150	50	10	1.2	1
19	200	300	100	50	10	1.2	1
20	200	300	150	50	10	1.8	2
21	300	400	150	200	10	1.2	2
22	200	400	100	50	20	1.2	2
23	200	400	150	200	10	1.8	1
24	300	400	100	200	20	1.2	1
25	250	350	125	125	15	1.5	1
26	300	300	100	50	20	1.2	2
27	300	300	100	50	10	1.8	2
28	300	300	150	200	20	1.2	1
29	200	300	150	50	20	1.2	2
30	200	300	100	200	10	1.8	1
31	200	400	150	50	20	1.8	1
32	300	400	100	200	10	1.8	1
33	300	300	150	50	20	1.8	1
34	300	400	100	50	10	1.2	1
35	250	350	125	125	15	1.5	2

The experimental runs were performed in two blocks of 16 trials each in such a way that no main effects or first order interactions were confounded with any hidden time effects (such as unscheduled equipment maintenance during the experiment). Three center points were added to the design to provide a check for model nonlinearity. The design matrix appears in Table 4.2.

The experimental sequence was randomized in order to avoid biases due to equipment aging during the experiment.

4.2.2 Second Phase - RSM Modeling Experiment

Analysis of the first stage of the experiment revealed significant nonlinearity in nearly all responses, which indicated the necessity of quadratic models. Also, none of the input factors were found to have a statistically insignificant effect on all of the responses of interest. Thus, none were omitted from the response surface models derived in the subsequent phase. In order to obtain these models, it was necessary to augment the data gathered with a second experiment that employed a Central Composite Circumscribed (CCC) Box-Wilson design (see Figure 4.1). In this design, the 2-level factorial "box" was enhanced by further replicated experiments at the center (to provide a direct measure of the equipment and measurement replication error) as well as symmetrically located "star" points [19].

Table 4.3: Additional "Star Point" Recipes for Box-Wilson Experiment

Run	Pressure	RF Power	CCl_4 Flow	*He* Flow	O_2 Flow	Electrode Gap
36	250	350	125	125	3	1.5
37	250	231	125	125	15	1.5
38	250	350	125	200	15	1.5
39	250	350	125	125	15	0.8
40	369	350	125	125	15	1.5
41	250	350	125	0	15	1.5
42	250	350	125	125	15	1.5
43	250	350	66	125	15	1.5
44	250	350	184	125	15	1.5
45	250	350	125	125	15	1.5
46	250	350	125	125	15	1.5
47	250	350	125	125	15	2.2
48	250	350	125	125	15	1.5
49	250	469	125	125	15	1.5
50	131	350	125	125	15	1.5
51	250	350	125	125	27	1.5
52	250	350	125	125	15	1.5
53	250	350	125	125	15	1.5

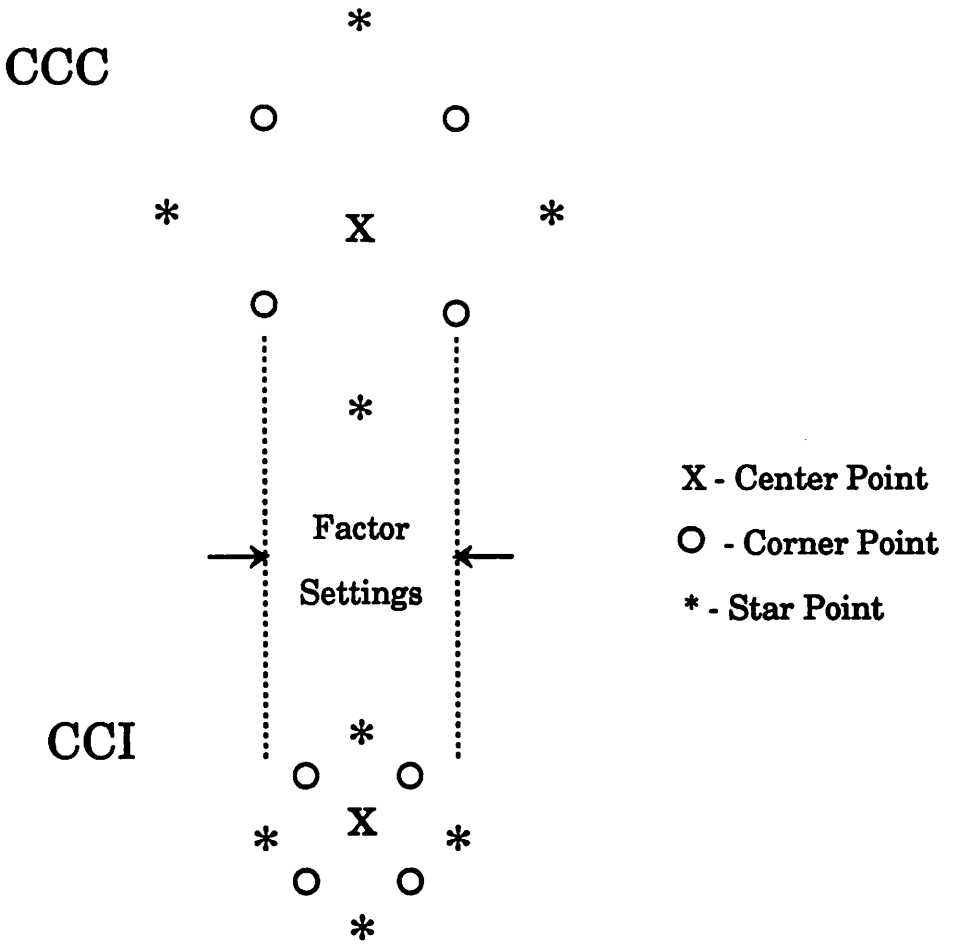


Figure 4.1 - Central composite Box-Wilson experimental designs [17].

A complete CCC design for six factors requires a total of 91 runs. In order to reduce the size of the experiment and combine it with the results from the screening phase, a half replicate design was again employed. The entire second phase required a total of 18 additional runs, whose recipes are shown in Table 4.3. A circumscribed design was selected as opposed to a inscribed (CCI) design, in order to allow the models to accurately predict the responses over the entire range of the input factor settings [17]. However, in the case of *He* flow for runs 39 and 41, the necessary star point required recipe settings (303 and -53 sccm) which are beyond the operational capabilities of the equipment. In this case, the recipe was modified to reflect the

maximum/minimum possible parameter settings of the Lam etcher (0 and 200 sccm, respectively). A graphic description of central composite designs appears in Figure 4.1.

4.3 Experimental Apparatus and Technique

4.3.1 Test Pattern Design

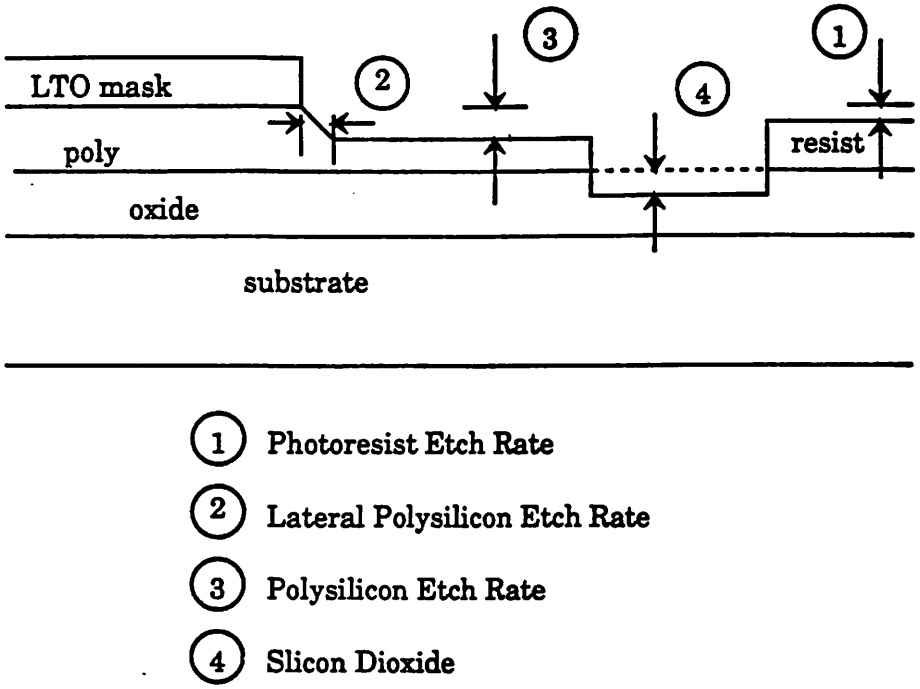


Figure 4.2 - Cross section of test structure describing the measurements of interest.

Etching was performed on a simple test structure designed to facilitate the simultaneous measurement of the vertical etch rates of polysilicon, SiO_2, and photoresist as well as the lateral etch rate of poly on the same wafer. This was done in order to insure that each of the models to be derived would be developed under identical process conditions. Preliminary investigations revealed that photoresist exhibited very poor etch selectivity when used as a mask for polysilicon. Due to this inadequacy, it was determined that low-temperature oxide (LTO) would serve as a much more effective mask for patterning the poly lines necessary for anisotropy measurements.

These lines were patterned using a simple 3-mask process which is described below. A cross section showing the critical measurement area is shown in Figure 4.2.

4.3.2 Test Pattern Fabrication

The patterns were fabricated on 4-in diameter silicon wafers. Approximately 1.2μm of phosphorus-doped polysilicon was deposited over 0.5μm of thermal SiO_2 by low-pressure chemical vapor deposition (LPCVD). The relatively thick layer of oxide was grown in order to prevent etching through the oxide by the less selective experimental recipes. Poly resistivity was measured at 86.0 Ω-cm. Oxide was grown in a steam ambient at 1000 °C. One micron of Kodak 820 photoresist was spun on and baked for 60 seconds at 120 °C. Poly lines for SEM photos were patterned with an LTO mask deposited at 450 °C by LPCVD.

4.3.3 Plasma Etch Equipment

The etching apparatus consisted of a Lam Research Corporation Autoetch 490 single-wafer parallel-plate system. The etching samples rest on the grounded lower electrode while the upper electrode is excited by a 13.56 MHz RF generator operating through a matching network. The anodized aluminum electrodes are circular and equal in area. The electrode walls are also composed of aluminum. Process gases are introduced into the chamber through nearly 1000 holes in the upper electrode in "showerhead" fashion. Reactor pressure is monitored with a capacitance manometer and controlled automatically with a throttle valve [20-21]. As mentioned above, the etcher was monitored via a real-time statistical process control scheme to ensure consistency in equipment operation throughout the experiment. A schematic diagram of the etching system appears in Figure 3.1 (see Chapter 3).

4.3.4 Measurement Methodology

Film thickness measurements were performed on five points per wafer (as in Figure 4.3) both before and after etching using a Nanometrics Nanospec AFT system in conjunction with an Alphastep 200 Automatic Step Profiler. Vertical etch rates were calculated by dividing the difference between the pre- and post-etch thickness by the etch time. The lateral etch rate for poly was determined via SEM photos by measuring the difference between the pre- and post-etch linewidth under the assumption that the pre-etch width was that at the base of the poly line (see Figure 4.17). Since the average polysilicon etch selectivity with respect to LTO was about 10:1, the contribution of the eroding LTO mask to the measured lateral rate was insignificant.

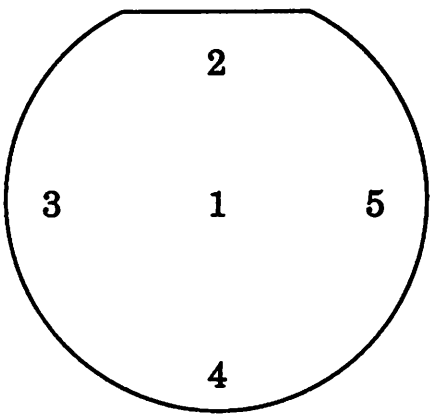


Figure 4.3 - Wafer Measurement Sites.

Expressions for the selectivity of the poly with respect to oxide (S_{ox}) and with respect to photoresist (S_{ph}) along with percent anisotropy (A) and percent nonuniformity (U), respectively, are given below:

$$S_{ox} = \frac{R_p}{R_{ox}} \tag{4.1}$$

$$S_{ph} = \frac{R_p}{R_{ph}} \tag{4.2}$$

$$A = \left[1 - \frac{L_p}{R_p}\right] \tag{4.3}$$

$$U = \frac{|R_{pc} - R_{pe}|}{R_{pc}} * 100 \tag{4.4}$$

where R_p is the mean vertical poly etch rate over the five points, R_{ox} is the mean oxide etch rate, R_{ph} is the mean resist etch rate, L_p is the lateral poly etch rate, R_{pc} is the poly etch rate at the center of the wafer, and R_{pe} is the mean poly etch rate of the four points located about one inch from the edge [1,22].

4.4 Experimental Results

Experimental data was analyzed using the *R/S Discover* commercial software package [17]. After the initial screening experiment, a few of the input factors were found to have an insignificant effect upon individual responses. However, no single factor was statistically irrelevant to all five responses of interest. For example, although the electrode gap spacing had little effect on the etch selectivity with respect to oxide, it had a dramatic impact on uniformity. Table 4.4 shows the significance level of the *student-t* statistic for each of the main effects.

Table 4.4: Results of Screening Experiment

Factor	Statistical Significance				
	R_p	S_{ox}	S_{ph}	U	A
Pressure	0.0090	0.0001	0.0001	0.0677	0.3008
RF Power	0.0001	0.0046	0.0001	0.0493	0.5119
CCl_4	0.0032	0.0410	0.0001	0.0672	0.5244
He	0.0001	0.0001	0.0001	0.0002	0.0157
O_2	0.0043	0.0669	0.0014	0.9581	0.6418
Gap	0.0185	0.4134	0.0001	0.0107	0.4634

** Only factors with a significance < 0.05 are considered significant.*

The above results indicate that all six controlled parameters have a significant effect both on etch rate and resist selectivity. On the other hand, oxide selectivity is impacted mostly by pressure, power, CCl_4 and helium flow. Etch uniformity depends primarily on power, helium flow and gap spacing. Finally, for the range of our experiment, anisotropy depends only on helium flow. The additional 18 runs of the second phase of the experiment yielded quadratic models which indicate the precise interaction between input factors and the four responses. These models are discussed below.

4.4.1 Polysilicon Etch Rate

Fitting a regression model for R_p yielded the following expression:

$$\begin{aligned} R_p = & -245 - 4.24P + 11.0Rf + 0.742CCl_4 + 11.2He + 523G \\ & + 35.9O_2 - 0.034P*He + 7.82P*G + 0.085Rf*CCl_4 - 8.36Rf*G \\ & - 0.132(CCl_4)^2 + 0.059CCl_4*He - 0.059He^2 \quad \pm 104\,\text{Å}\ /min \end{aligned} \tag{4.5}$$

where R_p is in angstroms/minute (Å /min) and the units of every other parameter are given in Table 4.1. This equation was derived by stepwise regression [23], and it will predict the mean response of the equipment with a one-sigma prediction error of ± 104 Å /min. The actual measured response of the equipment will vary around the mean value with a one-sigma replication error of ± 309 Å /min. The Analysis of Variance (ANOVA) table for the etch rate model is shown in Table 4.5.

The F-test shows that this model is highly significant, since the regression mean square, which is the amount of variation explained by the proposed model, is significant. This fact is verified by the F-ratio statistic. If the regression mean square was not significant, then this ratio would be distributed according to the F distribution with 15 and 37 degrees of freedom. The value 16.86, however, is highly unlikely to occur in the F(15,37) distribution. The lack-of-fit F-test reveals little evidence that the inclusion of additional terms would improve this model, since

a lack-of-fit F-ratio as large as 2.66 occurs 7.5% of the time in the F(29,8) distribution [7,19]. Therefore, most of the residual of the model originates from experimental error. The "adjusted R^{2}" is a parameter which measures the fraction of the total variation in the data accounted for by the model [11]. A scatterplot of the predicted etch rate values versus the corresponding experimental values is shown in Figure 4.4.

Table 4.5: ANOVA for Poly Etch Rate Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	52	24717141	475329.63		
Regression	13	20983554	1614120.00	16.86	0.000
Residual	39	3733587	95732.99		
Lack of Fit	31	3402778	109767.03	2.66	0.075
Error	8	330809	41351.11		

Adjusted $R^2 = 0.799$

Although this empirical etch rate model is fairly complex, a few interesting relationships emerge from the contour plots of Figures 4.5 and 4.6. In Figure 4.5, R_p surfaces are plotted against RF power and chamber pressure with all other parameters set at their nominal values. For high process throughput, etch rate should preferably be as high as possible. This occurs at high power and high pressure, since higher pressures provide a more suitable environment for chemical etching by radicals [1]. Alternatively, as can be seen in Figure 4.6, high etch rates occur when the gap is narrow and the flow rate is high. The observed relationship between the etch rate, power and the CCl_4 flow is consistent with the commonly held belief that the amount of etching is proportional to the adsorbed chlorine concentration on the polysilicon surface. This concentration is increased both by greater CCl_4 flow and high RF power, which enhances the rate of the following electron-impact dissociation and electron attachment reactions [13]:

$$CCl_4 + e^- \rightarrow CCl_3 + Cl + e^- \qquad (4.6)$$

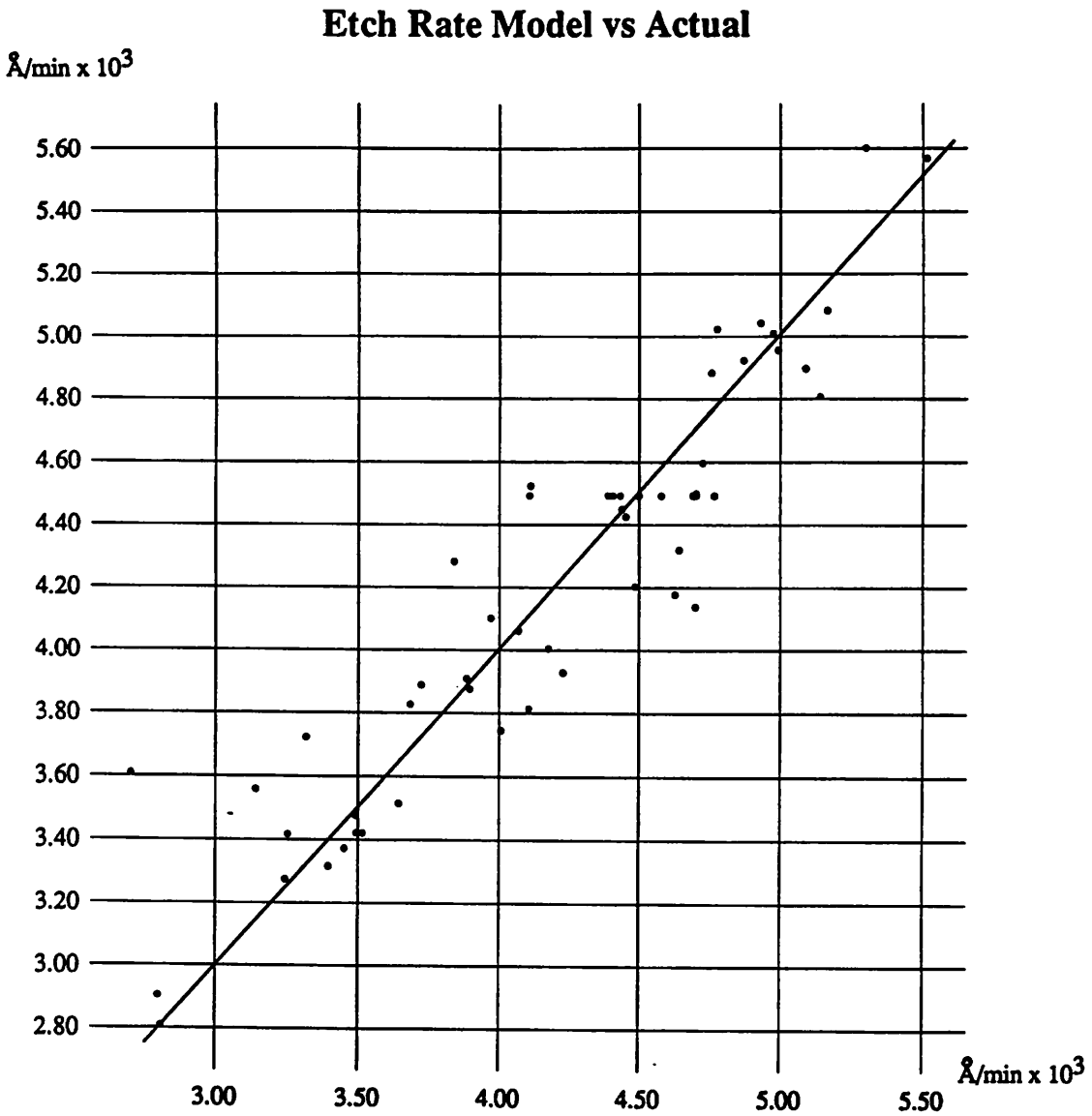


Figure 4.4 - Scatterplot of etch rates predicted by empirical model versus actual experimental values. The straight line represents the region of perfect agreement between model and experiment.

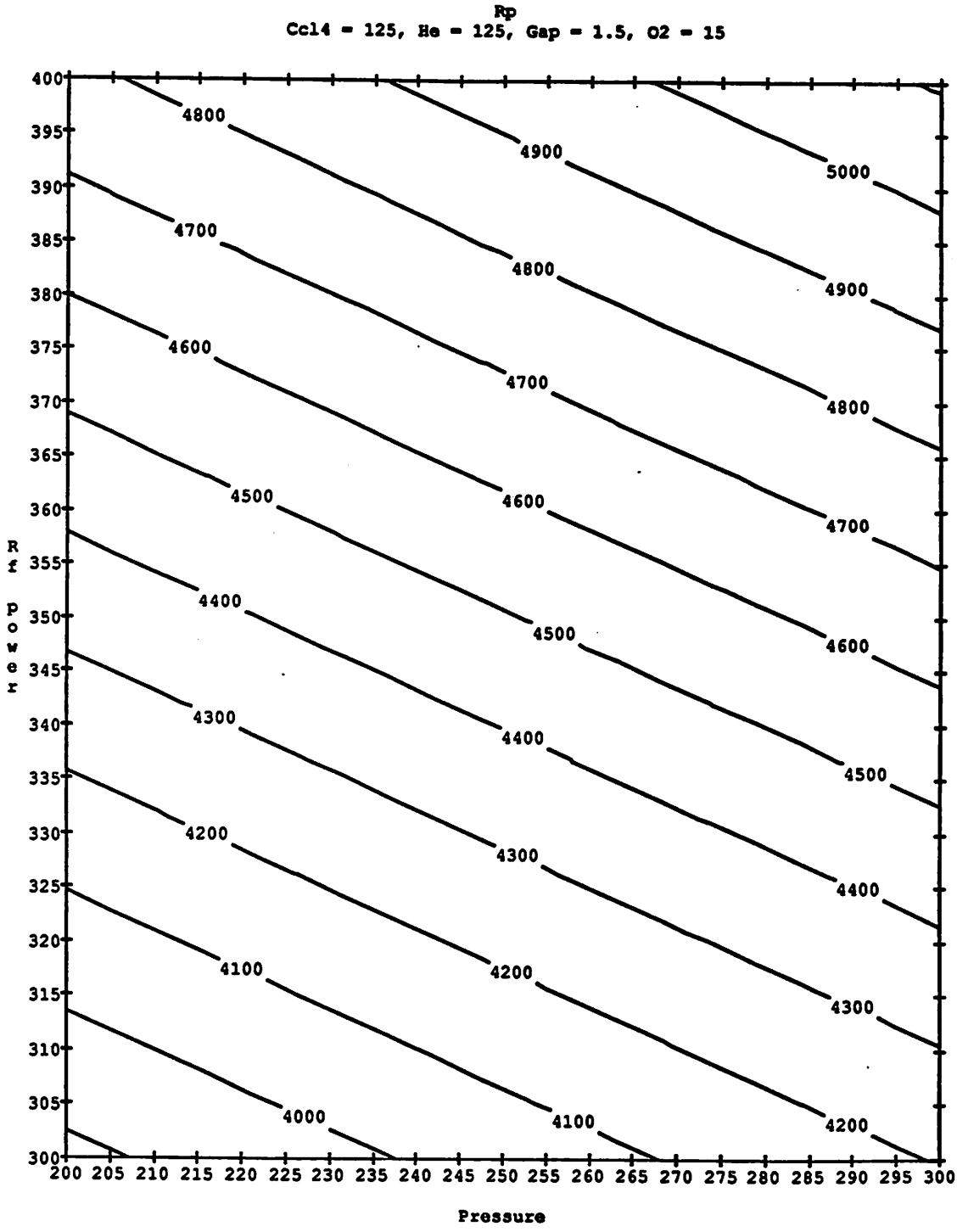


Figure 4.5 - Contour plot of polysilicon etch rate versus RF power and pressure.

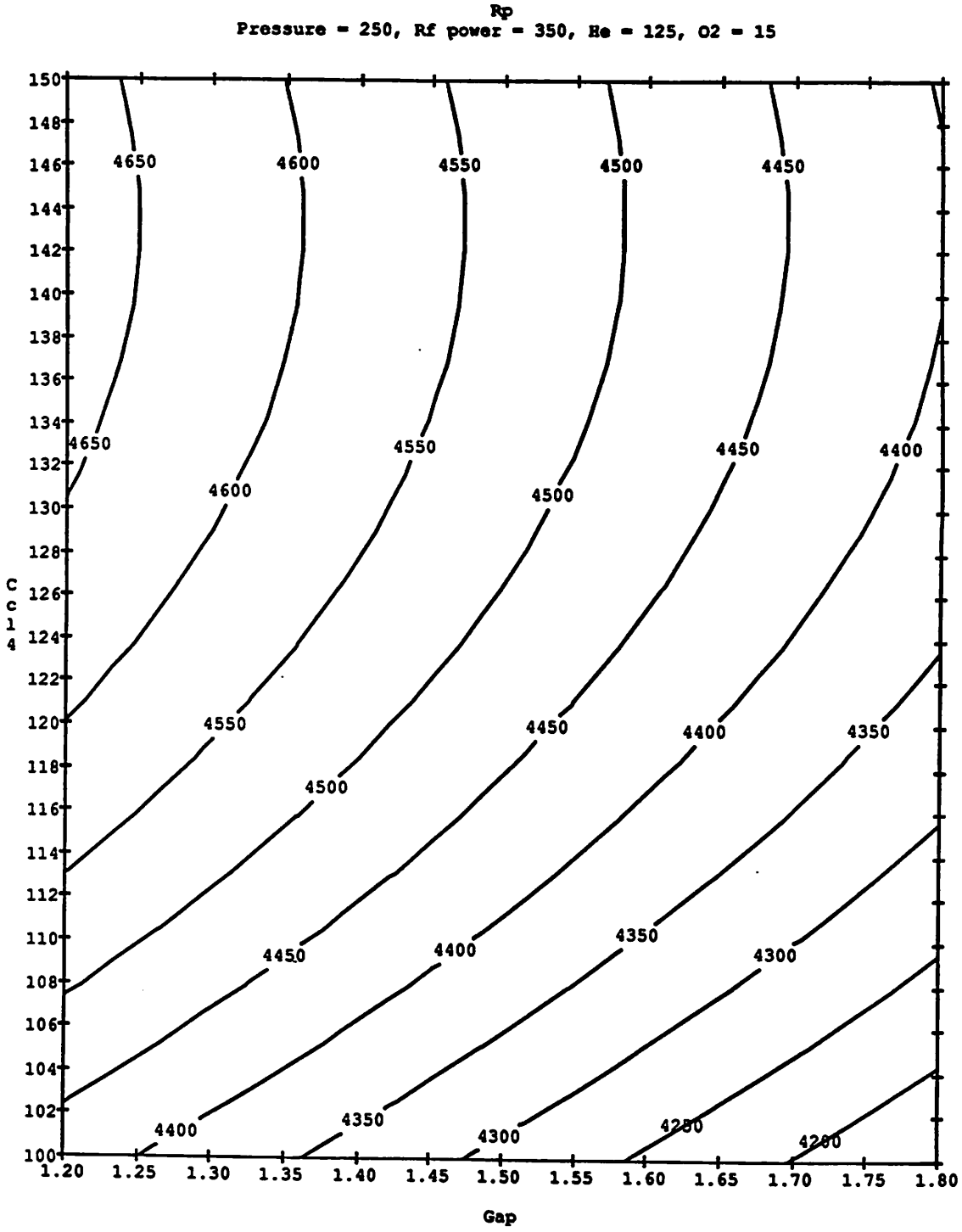


Figure 4.6 - Contour plot of polysilicon etch rate versus CCl_4 flow and electrode spacing.

$$CCl_4 + e^- \rightarrow CCl_2 + Cl^- + Cl \qquad (4.7)$$

However, it is shown below that these choices might compromise the other objectives of the etch step.

4.3.2 Etch Uniformity

The uniformity regression model and corresponding ANOVA table are:

$$U = -11.5 - 0.0385P + 0.0937Rf + 0.710CCl_4 - 0.415He - 8.90G \qquad (4.8)$$
$$-(1.77e{-}3)Rf{*}CCl_4 + (1.38e{-}3)Rf{*}He - (1.40e{-}3)CCl_4{*}He + (7.98e{-}4)He^2 \quad \pm 2.22[\%]$$

Table 4.6: ANOVA for Etch Uniformity Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	52	5896.02	113.39		
Regression	9	3987.96	443.11	9.99	0.000
Residual	43	1908.06	44.37		
Lack of Fit	35	1546.79	44.19	0.98	0.562
Error	8	361.27	45.16		

Adjusted $R^2 = 0.609$

The prediction error of the uniformity model is ± 2.22 (measured in %). The equipment replication error around the average value given by the model is ± 6.66 [%]. Tests for significance reveal that all model coefficients are relevant. In addition, there is no evidence for lack of fit. The scatterplot of predicted nonuniformity versus experimental data appears in Figure 4.7. The contours in Figures 4.8 and 4.9 depict the behavior of the model. In Figure 4.8, U is plotted against pressure and power. Optimum uniformity is observed at high pressure and low power. Thus, good uniformity is achieved at the expense of high etch rates. The effects of He flow and electrode spacing are observed in Figure 4.9. This plot verifies the initial assumption that helium enhances uniformity, but only up to a relatively low optimum value of He flow rate. Beyond this value, U begins to degrade.

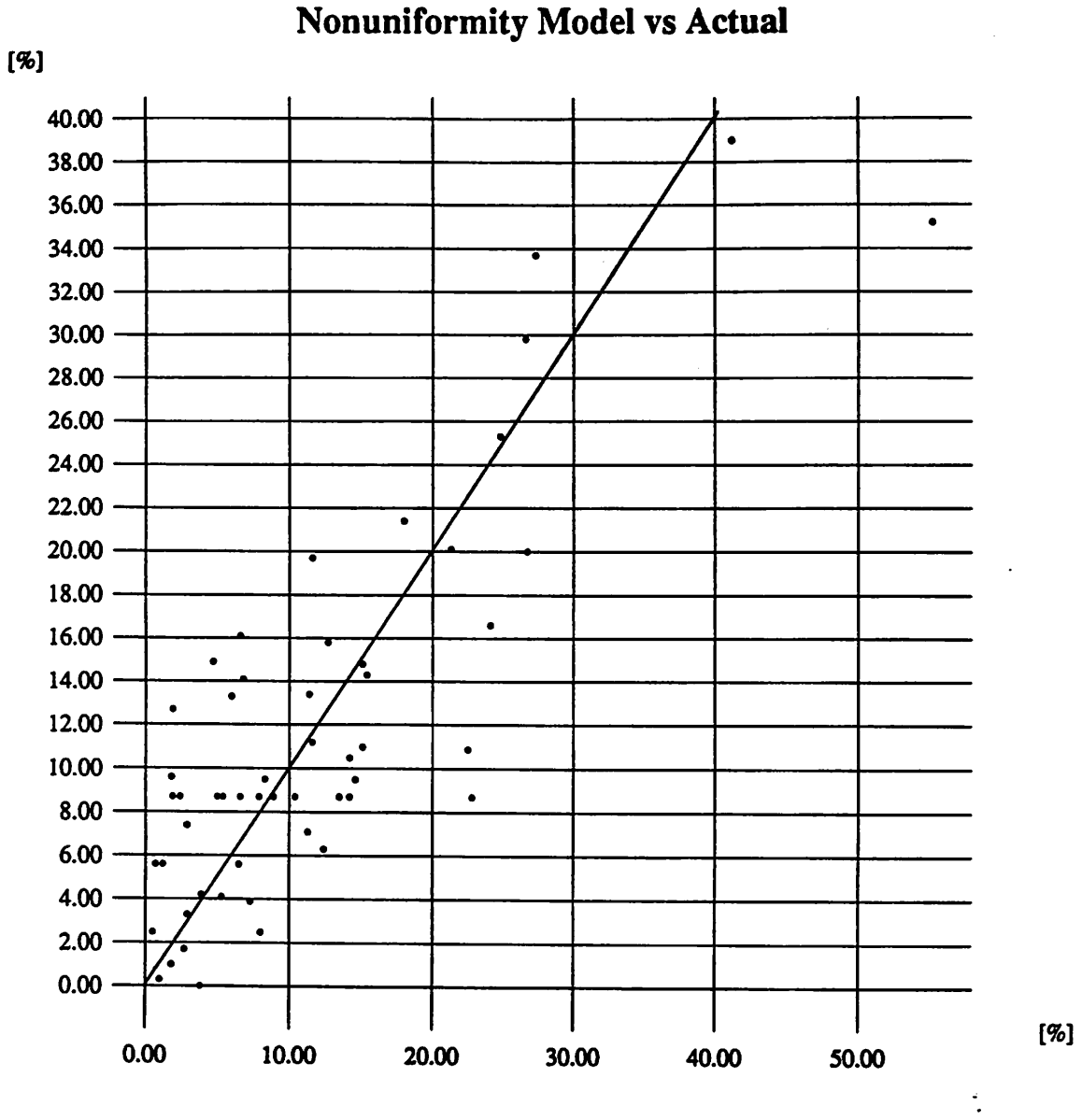


Figure 4.7 - Scatterplot of predicted nonuniformity versus experimental values.

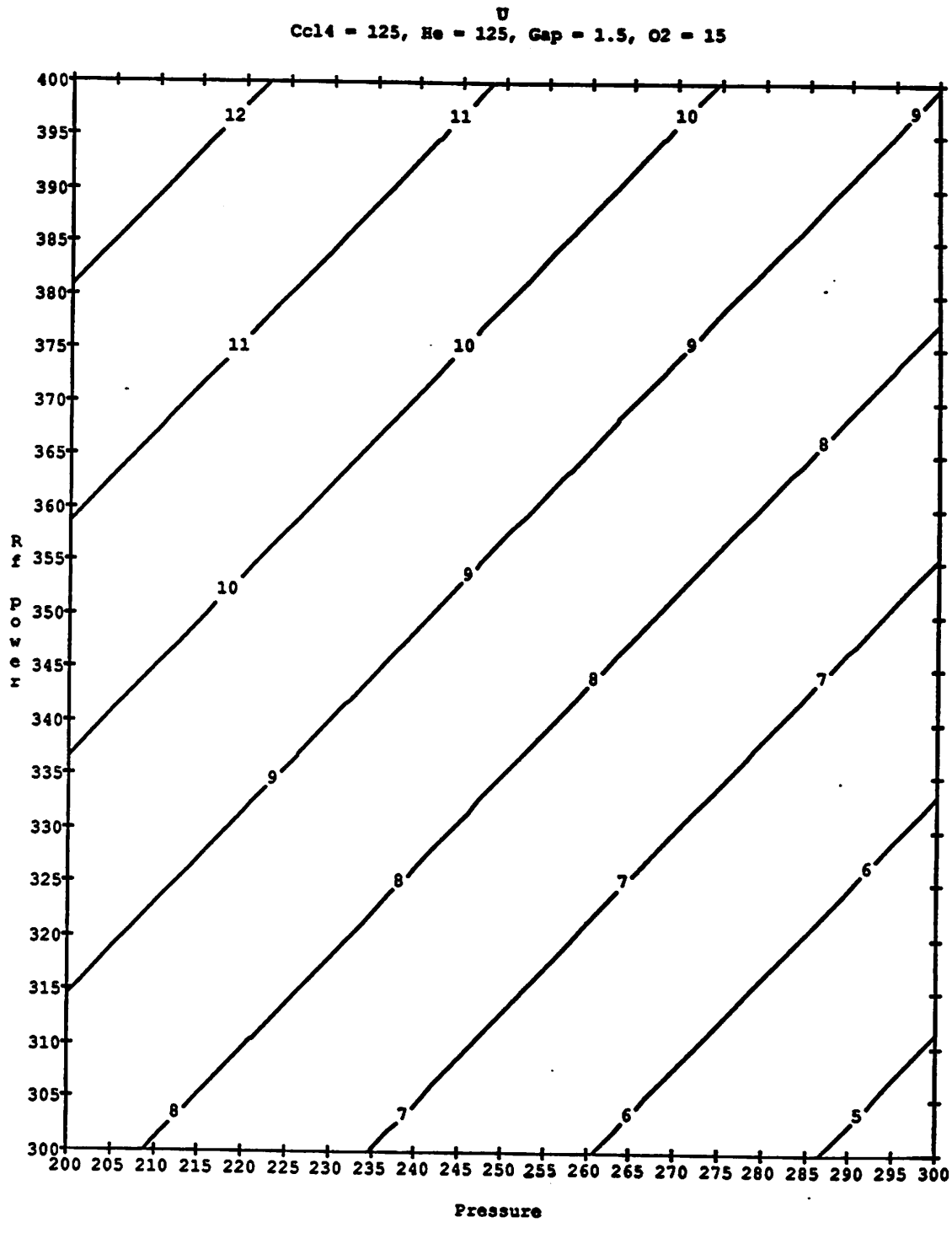


Figure 4.8 - Contour plot of etch uniformity versus RF power and pressure.

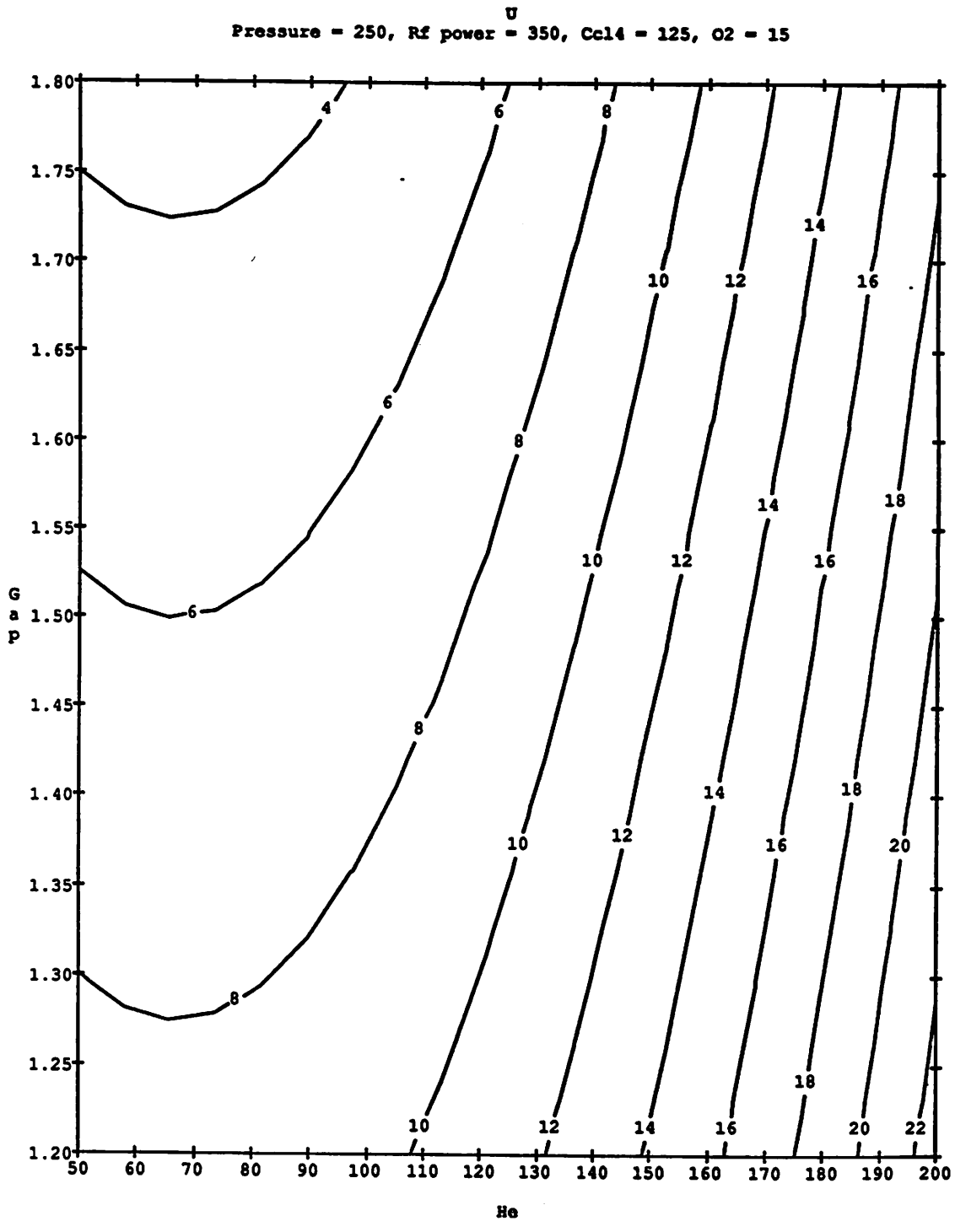


Figure 4.9 - Contour plot of etch uniformity versus electrode spacing and *He* flow.

4.4.3 Oxide Selectivity

The regression model and ANOVA table for S_{ox} are given below:

$$S_{ox} = -13.1 + 0.097P + 0.04Rf - 0.06CCl_4 - 0.059He + 0.079O_2 \\ -(2e{-}4)P{*}Rf + (2.9e{-}4)P{*}CCl_4 - (3e{-}4)P{*}He \quad \pm 0.31 \tag{4.9}$$

Table 4.7: ANOVA for Oxide Selectivity Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	52	248.70	4.78		
Regression	9	213.26	23.70	28.76	0.000
Residual	43	35.43	0.82		
Lack of Fit	35	31.35	0.90	1.75	0.205
Error	8	4.09	0.51		

Adjusted $R^2 = 0.828$

The one-sigma prediction error of this model is ± 0.31, and the replication error inherent in the equipment is ± 0.91. The F-tests reveal that the overall model is highly significant, and that there is no evidence that a more complex model is required. The scatterplot for the oxide selectivity model appears in Figure 4.10. A few implications of the oxide selectivity model are seen in Figures 4.11 and 4.12. Figure 4.11 shows S_{ox} contours versus RF power and pressure. According to this plot, highest oxide selectivity occurs at high pressure and low power. These results reflect the fact that high ion energies (produced by high RF power) generally tend to degrade etch selectivity [1]. Thus, a trade-off exists between high etch rate and good selectivity in terms of power. The effects of CCl_4 flow and pressure can be seen in Figure 4.12. The highest oxide selectivity occurs when pressure and CCl_4 flow are both high.

4.4.4 Photoresist Selectivity

The regression model and ANOVA table for S_{ph} are:

$$S_{ph} = 7.56 + 0.009P + 0.014Rf - 0.022CCl_4 + 0.006He - 2.59G - 0.099O_2 \\ -(5e{-}5)P{*}Rf + (1.3e{-}4)P{*}CCl_4 - (7e{-}5)P{*}He + (3.7e{-}4)P{*}O_2 \tag{4.10}$$

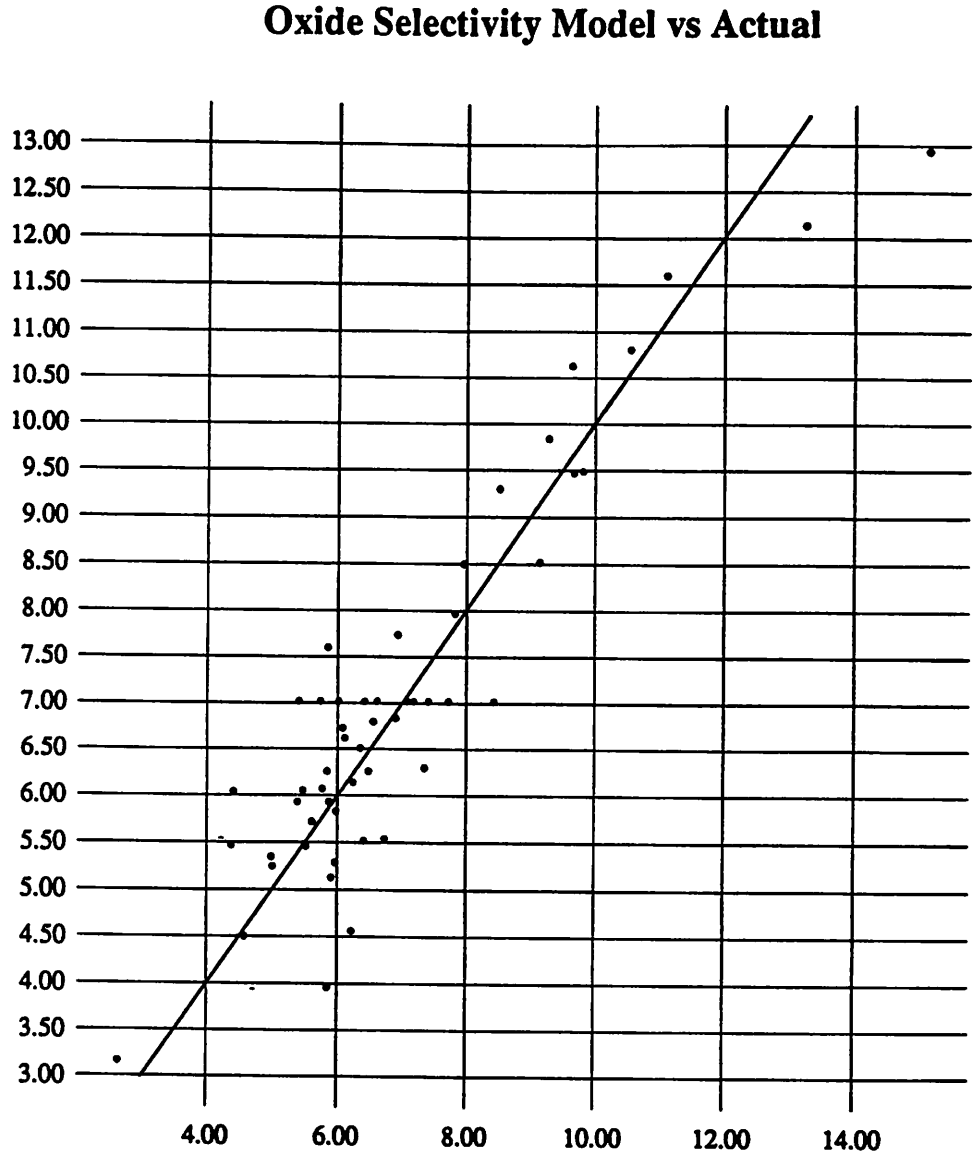


Figure 4.10 - Scatterplot of predicted oxide selectivity versus experimental values.

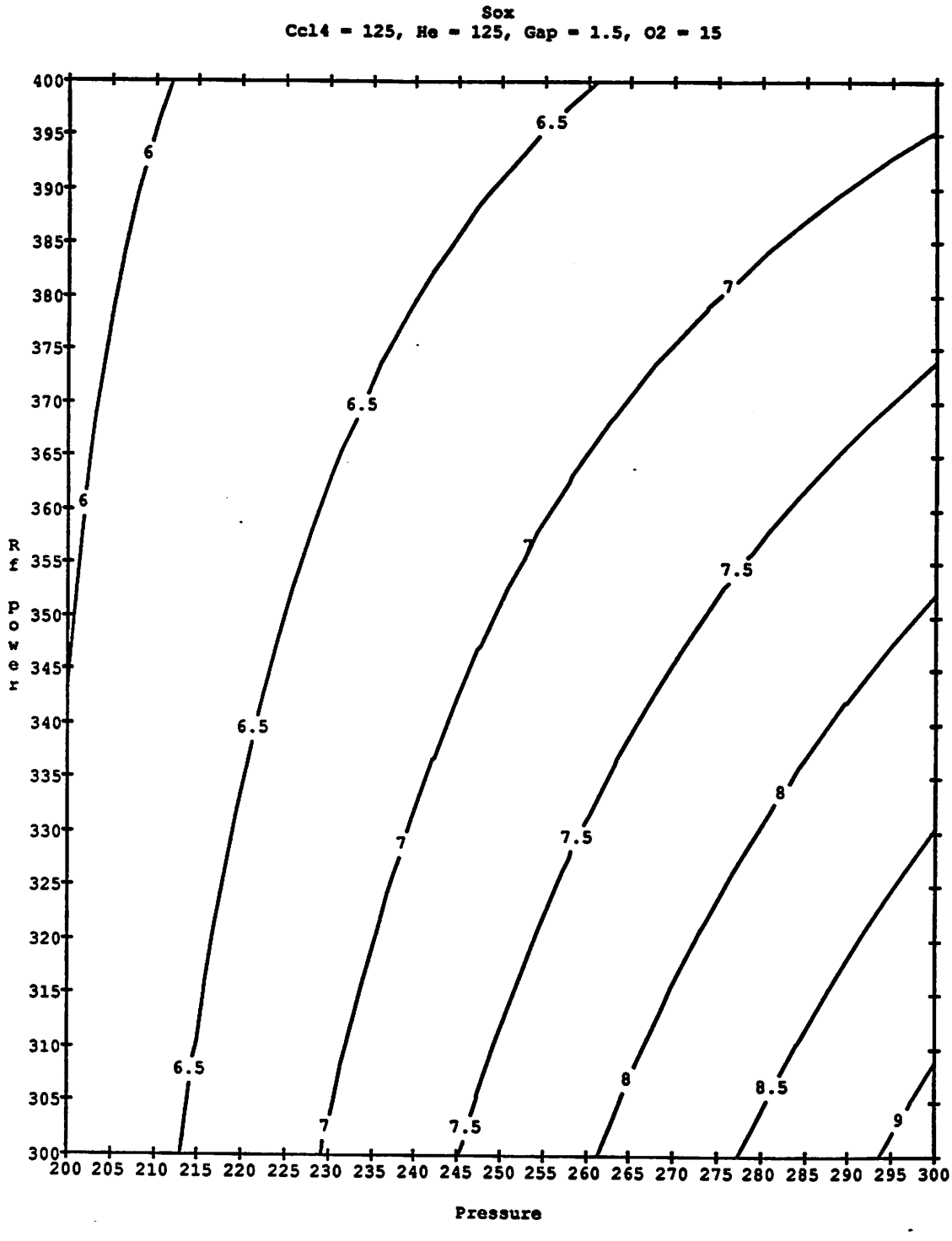


Figure 4.11 - Contour plot of oxide selectivity versus RF power and pressure.

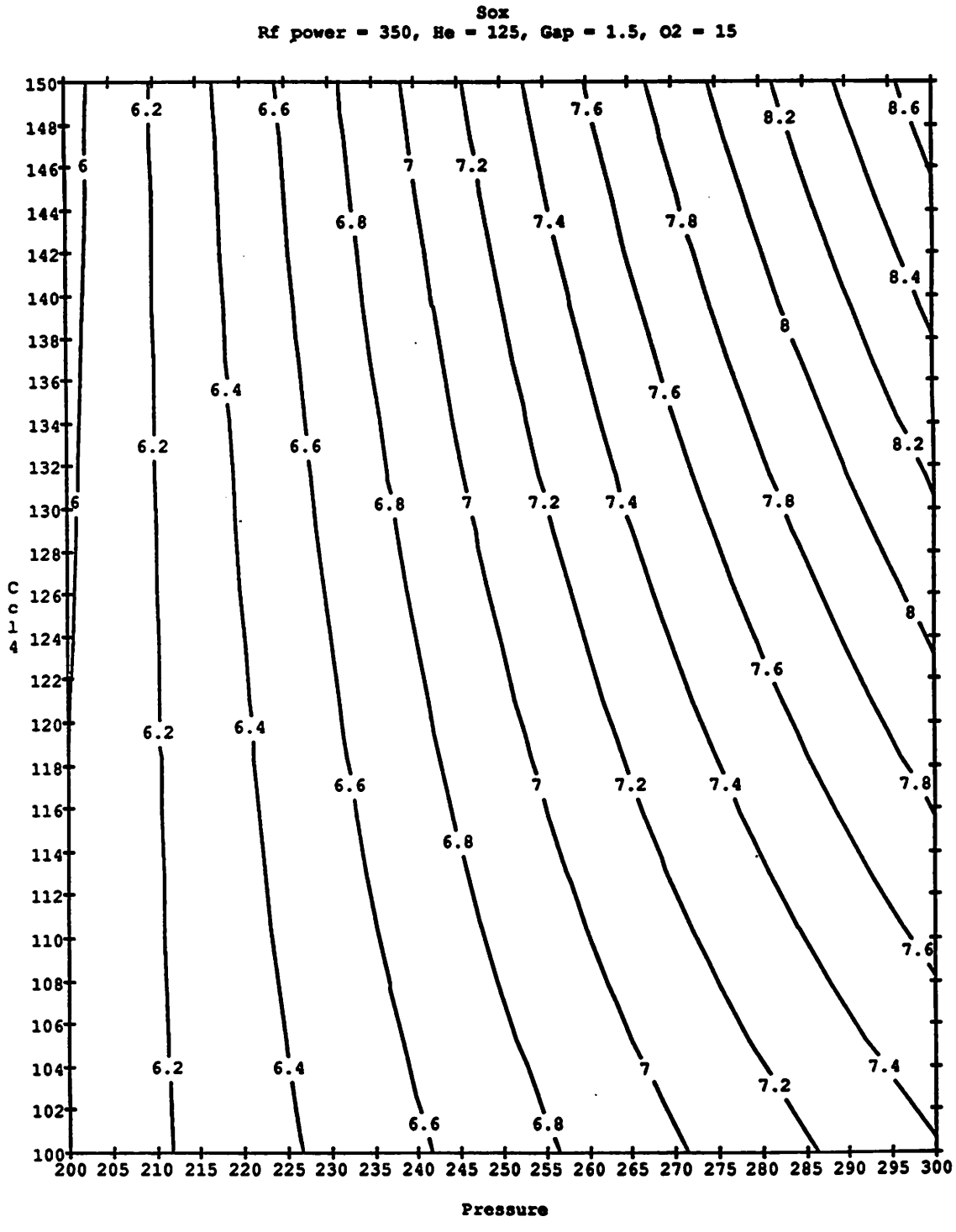


Figure 4.12 - Contour plot of oxide selectivity versus CCl_4 flow and pressure.

$$+ (2.7e{-}5)Rf^2 + (3.6e{-}5)Rf{*}He - (5e{-}5)CCl_4{*}He + 0.757G^2 \quad \pm 0.09$$

Table 4.8: ANOVA for Photoresist Selectivity Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	52	15.24	0.29		
Regression	14	12.61	0.90	13.02	0.000
Residual	38	2.63	0.07		
Lack of Fit	30	2.42	0.08	3.07	0.050
Error	8	0.21	0.03		

Adjusted $R^2 = 0.764$

The one-sigma prediction error of this model is ± 0.09, and the equipment replication error is ± 0.95. Statistical tests for model complexity and fit give no reason to question the significance and adequacy of the resist selectivity model. The scatterplot for the resist selectivity model is shown in Figure 4.13. Figure 4.14 shows S_{ph} contours versus power and pressure, and Figure 4.15 shows the effects of CCl_4 flow and pressure. These plots indicate that photoresist selectivity possesses similar trends to that of oxide. This result is not surprising, since both oxide and resist are etched mechanically rather than chemically within the plasma [15].

4.4.5 Anisotropy

Early studies have reported significant undercutting of the etch mask during polysilicon plasma etching in CCl_4 [15]. In this study however, our primary interest was to derive models that describe equipment behavior in commonly used ranges of settings. This meant that the chosen range of RF power was high enough and the range of pressure was low enough such that the ions in the plasma struck the wafer surface almost exclusively at normal incidence. This type of ion bombardment preferentially accelerates the surface chemical reaction in the vertical direction, thereby producing highly anisotropic profiles [1]. Consequently, analysis of the etch anisotropy data from the first block of the screening phase of the experiment showed very anisotropic poly lines for all recipe variations. Scanning Electron Microscopy (SEM) photographs were used

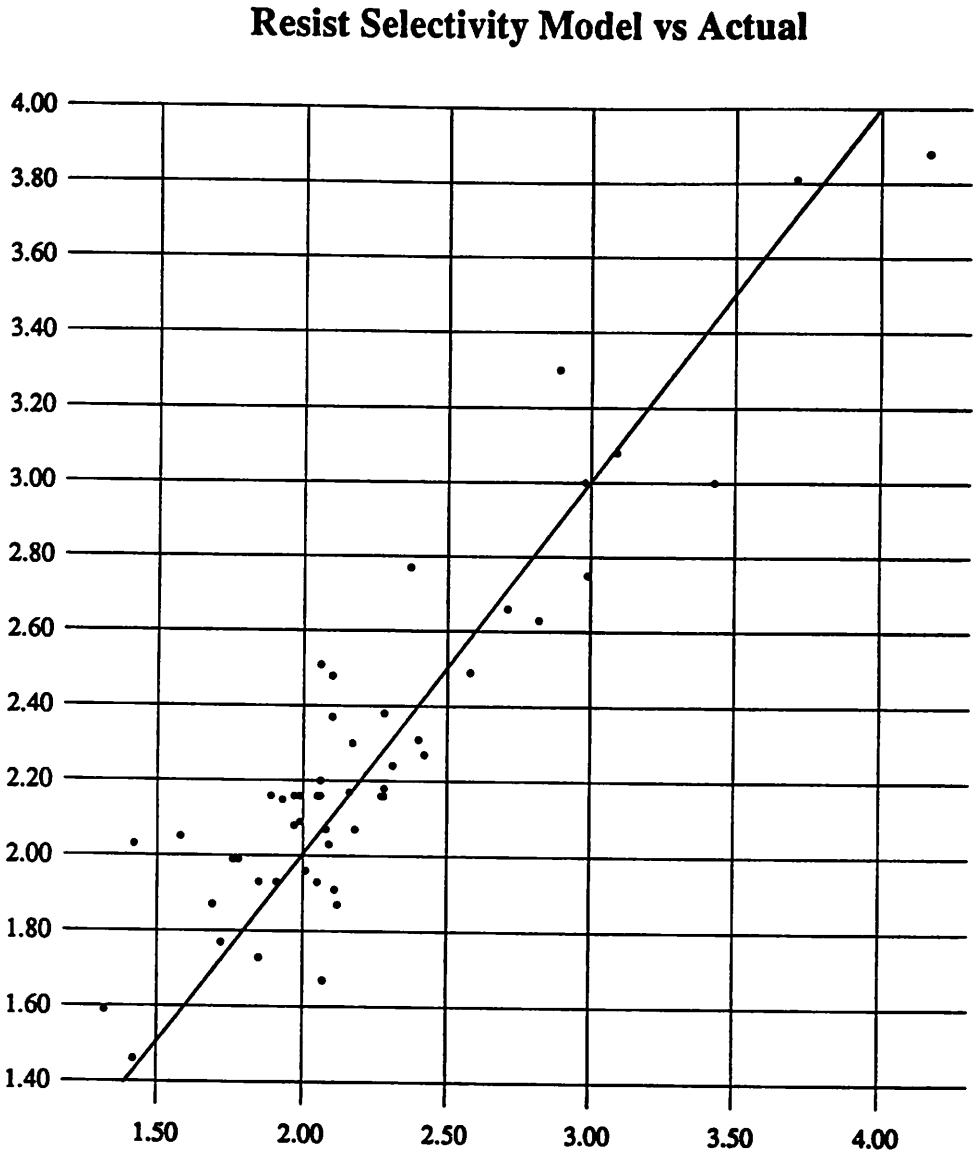


Figure 4.13 - Scatterplot of predicted resist selectivity versus experimental values.

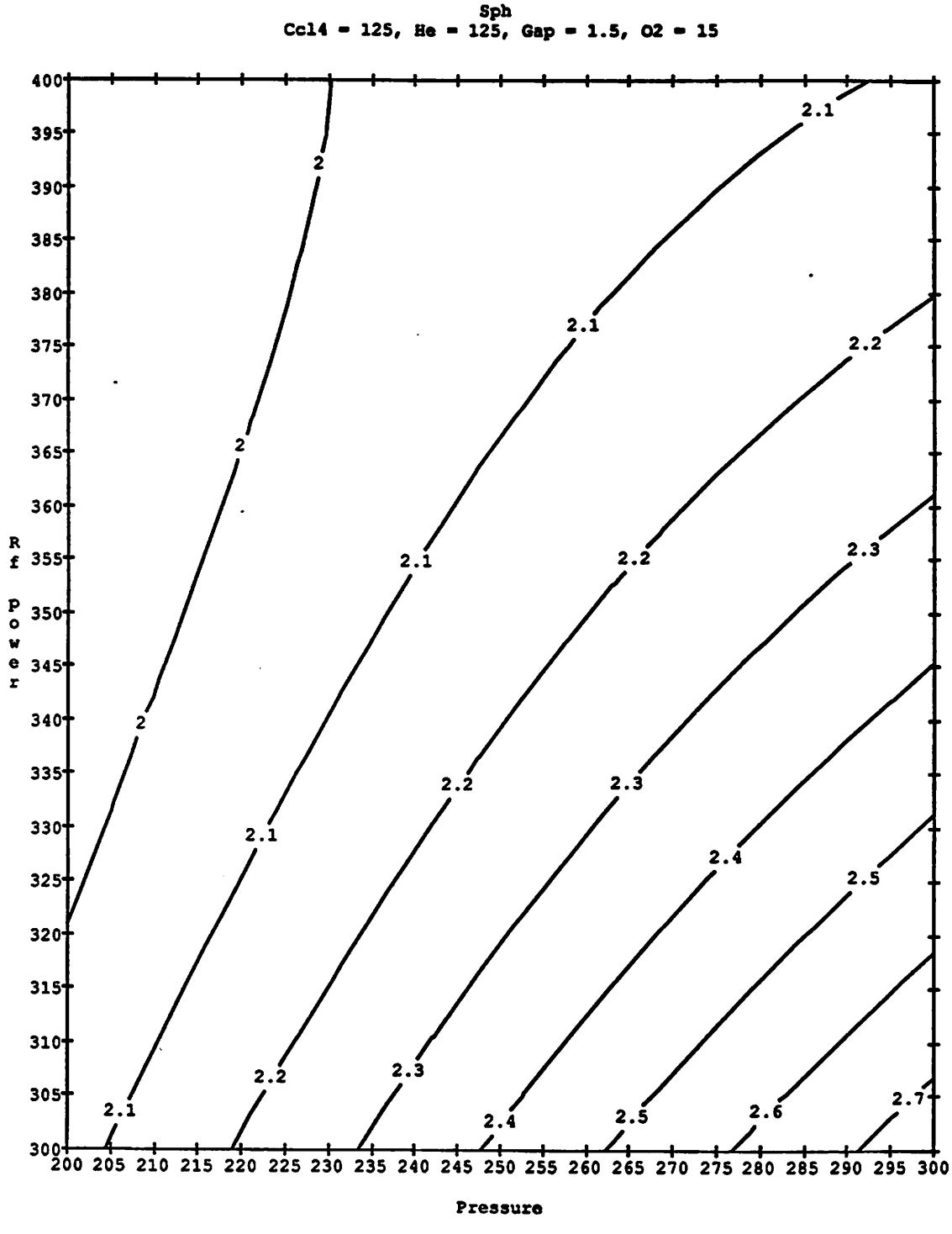


Figure 4.14 - Contour plot of resist selectivity versus RF power and pressure.

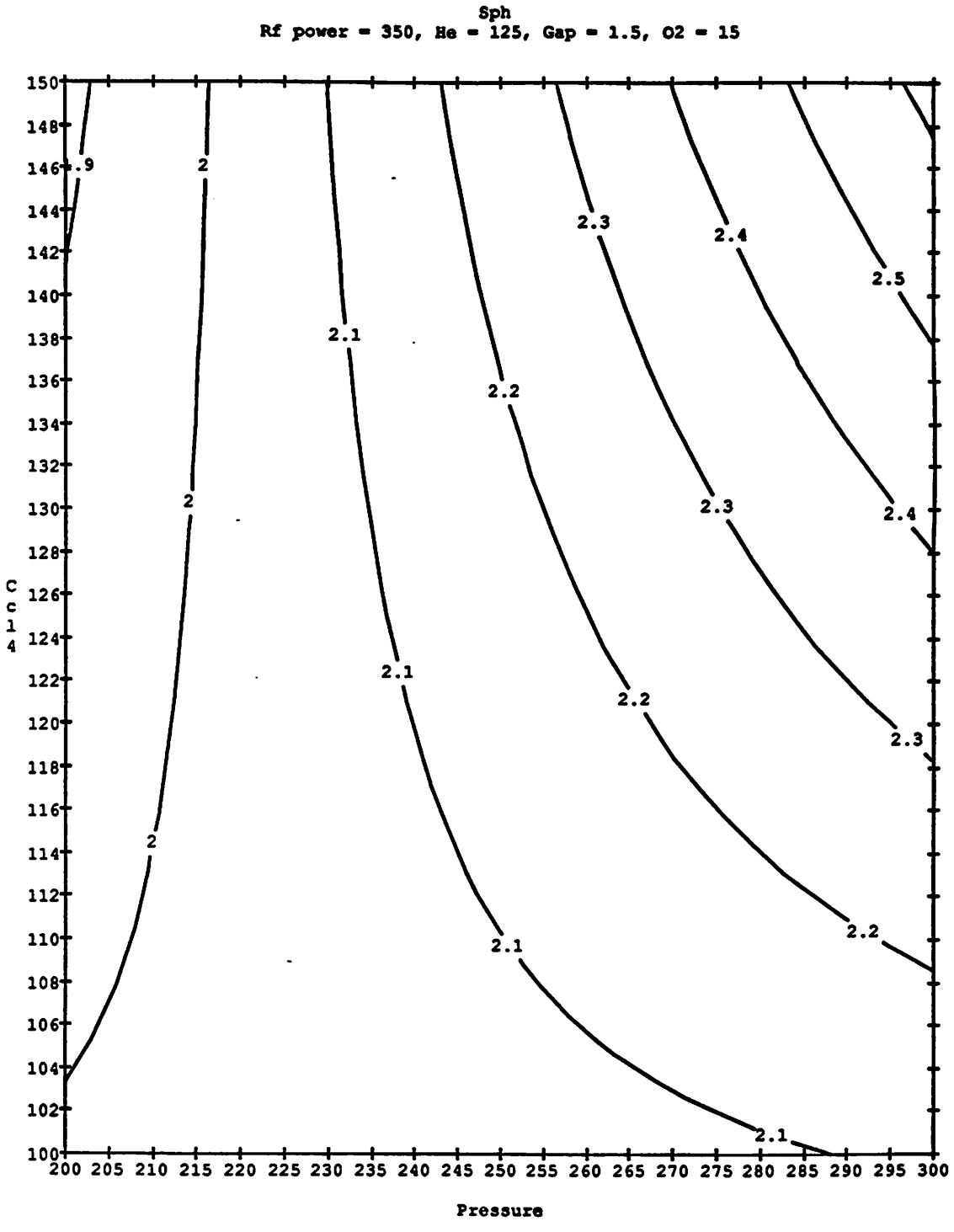


Figure 4.15 - Contour plot of resist selectivity versus CCl_4 flow and pressure.

to measure the sidewall slope as defined in Eq. (3). Because of the cost associated with SEM studies, only 18 of the first-phase wafers were examined. These samples were chosen according to a Resolution IV, 2^{6-2} fractional factorial design with three center points. This data exhibited no significant nonlinearity and rather high equipment and measurement replication error. Still, a simple, statistically significant anisotropy model was derived using only these trials. The model is given below:

$$A = 94.6 - 0.031He \quad \pm 1.82(\%) \tag{4.11}$$

The one-sigma replication error of this model is ± 3.24 (%). Anisotropy depends primarily upon helium flow for this range of input factors. As indicated in Figure 4.16, high anisotropy is obtained by reducing helium flow as much as possible. An SEM photo exhibiting the lateral etching of the polysilicon lines appears in Figure 4.17. As shown in Table 4.9, analysis of variance for this model revealed no evidence that the model is inadequate or any indication of lack of fit. Although this model is statistically significant, its high prediction and replication error imply that benefits from *He* adjustment can only be realized in a long production run.

Table 4.9: ANOVA for Anisotropy Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	18	269.83	14.99		
Regression	1	91.23	91.23	8.68	0.009
Residual	17	178.60	10.51		
Lack of Fit	16	178.35	11.15	45.50	0.116
Error	8	0.21	0.03		

Adjusted $R^2 = 0.299$

4.5 Model Verification

The accuracy of the above models was verified in two separate procedures. In the first, an experiment was conducted to optimize the standard etch recipe using numerical optimization

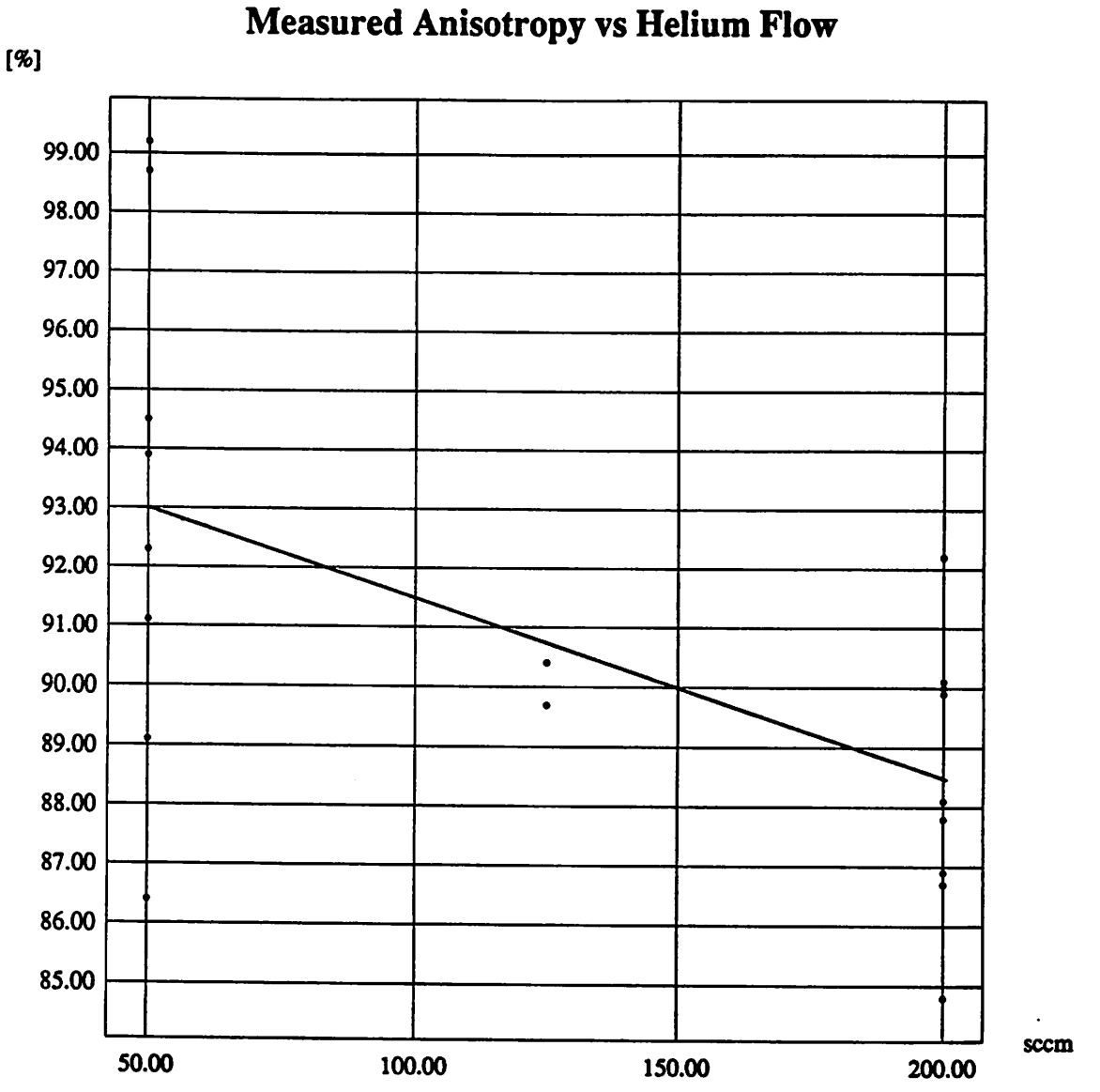
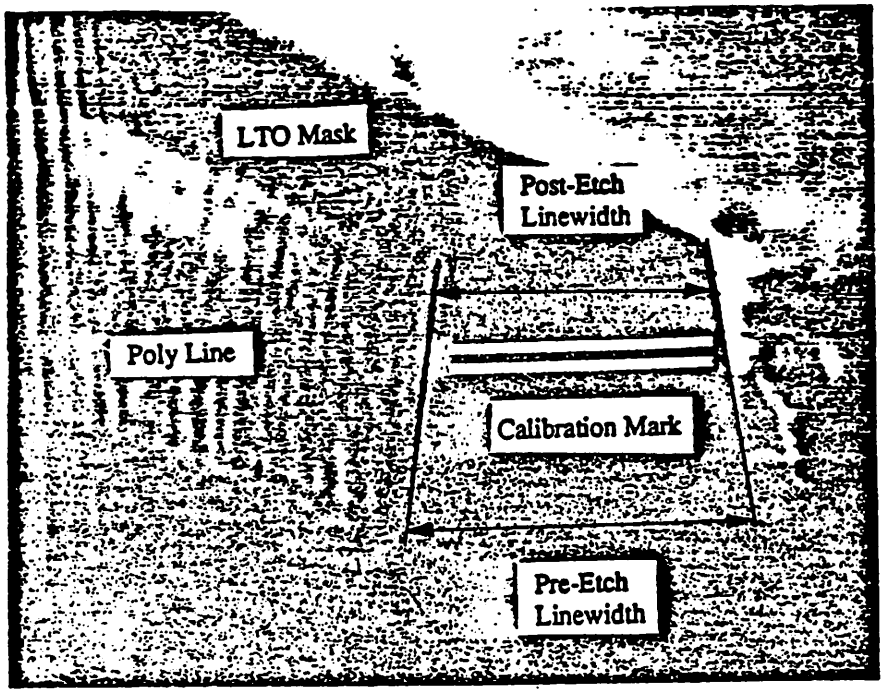
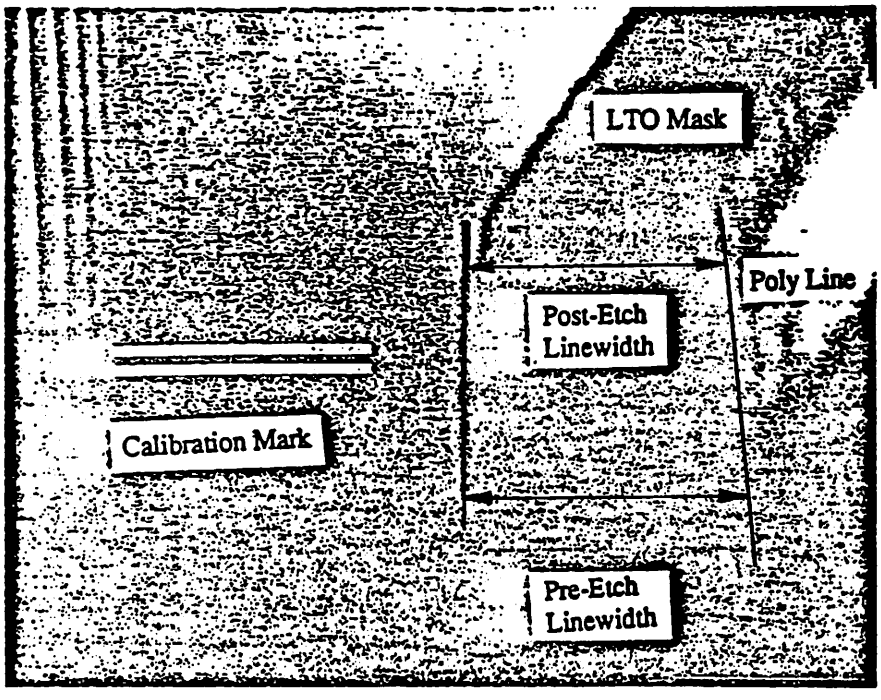


Figure 4.16 - Plot of anisotropy versus helium flow. The line represents the empirical model and the points correspond to actual data.



(a)



(b)

Figure 4.17 - SEM photos of typical polysilicon lines used to determine the sidewall slope and lateral etch rate for the anisotropy calculation. (a) is a line etched with a high *He* flow rate, and (b) was etched with a low *He* flow rate.

techniques along with the five empirical models. In the second, the models were used to predict the etch rate and anisotropy of various recipes designed to etch annealed polysilicon. In each case, the utility and precision of the etch models were corroborated by the experimental results.

4.5.1 Process Optimization

The etch models have been used to design a new etch recipe which exhibited improvement in all etch responses. The optimum recipe was determined using the Han Powell constraint optimization algorithm [25]. This recipe was designed to simultaneously increase etch rate, selectivities and anisotropy while minimizing nonuniformity. A comparison between the standard recipe and the optimized recipe appears in Table 4.10.

Table 4.10: Standard and Optimized Etch Recipes

Parameter	Standard Recipe	Optimized Recipe
RF Power (watts)	300	300
Pressure (mtorr)	280	300
Electrode Spacing (cm)	1.5	1.2
CCl_4 Flow (sccm)	130	150
He Flow (sccm)	130	50
O_2 Flow (sccm)	15	20

After the optimum recipe was determined, an experiment was undertaken to confirm the improvement of the etch responses. In this experiment, a total of six wafers were identically prepared according to the previously described process flow. Prior to the final etch step, these wafers were divided into two equal groups, one group undergoing the standard etch recipe and the other receiving the optimized treatment. The results of this experiment are summarized in Table 4.11. The last two columns in this table show the percent improvement in the etch response derived from optimizing the recipe and the statistical significance of this improvement based on

the *student-t* statistic [19]. Notably, significant improvement was obtained in nearly every case.

Table 4.11: Standard and Optimized Responses

Response	Std	Opt	% Change	Sign.
Etch Rate (Å /min)	3660	4467	22.0	0.03
Nonuniformity (%)	10.66	10.09	-5.3	0.56
Oxide Selectivity	9.58	20.10	109.8	0.01
Resist Selectivity	2.99	5.07	69.6	0.00

4.5.2 Using Models for Prediction

The empirical etch models which have been developed enable a process engineer to accurately predict etch responses over a wide range of etch recipes. This fact was verified by an experiment designed to characterize the plasma etching of annealed polysilicon using various recipes. In this procedure, the particular responses of interest were the etch rate and anisotropy. The results of this experiment are summarized below in Tables 4.12 and 4.13. In these tables, the difference between predicted and actual values is given in standard deviations as given by the equipment replication error (σ_R). (Note that anisotropy measurements were not made for recipes 2, 6, and 7).

Table 4.12: Etch Rate Predictions versus Actual Results

Recipe	Predicted Rate (Å /min)	Actual Rate (Å /min)	Difference (σ_R)
1	4161	3992	0.6
2	4161	4516	1.2
3	3973	3685	0.9
4	4161	4012	0.5
5	4348	4696	1.1
6	4348	4669	1.0
7	4348	4452	0.3

Table 4.13: Anisotropy Predictions versus Actual Results

Recipe	Predicted Anisotropy (%)	Actual Anisotropy (%)	Difference (σ_R)
1	93.1	95.0	0.6
3	90.6	90.0	0.2
4	93.1	89.0	1.3
5	92.1	91.5	0.2

In each of the above cases, the models predicted the etch behavior well within $\pm 1.96\sigma_R$ limits, thereby validating their accuracy at the 5% level of significance [19].

4.6 Summary

An economical two-phase experiment has been designed and conducted to characterize the etch rate, uniformity, selectivity to SiO_2 and photoresist, and anisotropy of n^+-doped polysilicon versus a comprehensive set of controlling parameters. These parameters were fit to quadratic response surface models.

Undoubtedly, some of the second order effects reported here apply only to the specific apparatus used for our experimental processing. Additional characterization effort has to be undertaken in order to apply this methodology to other production-worthy plasma etchers. On the other hand, it has been shown that these models describe the operation of the characterized equipment very precisely. These models can also be augmented to reflect equipment aging by using on-line experimentation [24]. Further, since statistical estimates of both prediction and replication errors were derived, these models can be used for the long range optimization of the etching process. This consideration is extremely important for long production runs.

Unlike computationally expensive physically-based simulators which are often impractical due to their slowness and lack of precision, the empirical models derived herein can be used for a variety of manufacturing purposes, including diagnosis, recipe generation, and statistical process

control. This has been accomplished by organizing and storing the models in an object-oriented software library (see Appendix 4.2). In the next chapter, the specific use of the models for malfunction diagnosis will be described in greater detail.

References for Chapter 4

[1] D. M. Manos and D. L. Flamm, *Plasma Etching: An Introduction*, San Diego: Academic Press, 1989.

[2] T. J. Cotler, M. S. Barnes, and M. E. Elta, "A Monte Carlo Microtopography Model for Investigating Plasma/Reactive Ion Etch Profile Evolution," *Journal of Vacuum Science & Technology B*, vol. 6, no. 2, March/April, 1988.

[3] M. S. Barnes, T. J. Cotler, and M. E. Elta, "Large-Signal Time-Domain Modeling of Low-Pressure RF Glow Discharges," *Journal of Applied Physics*, vol. 61, no. 1, January, 1987.

[4] A. P. Paranjpe, J. P. McVittie, and S. A. Self, "Numerical Simulation of 13.56 MHz Symmetric Parallel Plate RF Glow Discharges in Argon," *Proceedings of the 41st Gaseous Electronics Conference*, October, 1988.

[5] D. B. Graves, "Modeling Plasma-Enhanced CVD Reactors for Semiconductor Fabrication," *Short Course on Chemical Vapor Deposition*, University Extension, UC-Berkeley, August 7-9, 1989.

[6] G. E. P. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces*, New York: Wiley, 1987.

[7] K. K. Low and S. W. Director, "An Efficient Methodology for Building Macromodels of IC Fabrication Process," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 12, December, 1989.

[8] P. C. Karulkar and M. A. Wirzbicki, "Characterization of Etching of Silicon Dioxide and Photoresist in a Fluorocarbon Plasma," *Journal of Vacuum Science & Technology B*, vol. 6, no. 5, Sep/Oct, 1988.

[9] P. E. Riley, A. P. Turley, and W. J. Malkowski, "Development of a Multistep SiO_2 Plasma Etching Process in a Minibatch Reactor Using Response-Surface Methodology," *Journal of the Electrochemical Society*, vol. 136, no. 4, April, 1989.

[10] P. E. Riley, "Development of a Highly Uniform Silicon Dioxide Etching Process Using Response-Surface Methodology," *Journal of the Electrochemical Society*, vol. 133, no. 9, September, 1986.

[11] M. W. Jenkins, M. T. Mocella, K. D. Allen, and H. H. Sawin, "Modeling Plasma Etching Processes Using Response Surface Methodology," *Solid State Technology*, April, 1986.

[12] B. E. Thompson and H. H. Sawin, "Polysilicon Etching in SF_6 RF Discharges," *Journal of the Electrochemical Society*, vol. 133, no. 9, September, 1986.

[13] E. Gogolides and H. H. Sawin, "n^+-Polysilicon Etching in CCl_4/He Discharges: Characterization and Modeling," *Journal of the Electrochemical Society*, vol. 136, no. 4, April, 1989.

[14] P. E. Riley and D. A. Hanson, "Study of Etch Rate Characteristics of SF_6/He Plasmas by Response-Surface Methodology: Effects of Interelectrode Spacing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 2, no. 4, November, 1989.

[15] D. H. Bower, "Planar Plasma Etching of Polysilicon Using CCl_4 and NF_3," *Journal of the Electrochemical Society*, vol. 129, no. 4, April, 1982.

[16] C. B. Zarowin and R. S. Horwath, "Control of Plasma Etch Profiles with Plasma Sheath Electric Field and RF Power Density," *Journal of the Electrochemical Society*, vol. 129, no. 11, November, 1982.

[17] *RS/Discover User's Guide*, BBN Software Products Corporation, June 1988.

[18] N. H. Chang, "Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB/ERL M90/61*, May, 1990.

[19] G. E. P. Box, W. B. Hunter, and J. S. Hunter, *Statistics for Experimenters*, New York: Wiley, 1978.

[20] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, March, 1985.

[21] "Advanced Dry Etching of Aluminum and its Alloys," *Solid State Technology*, April, 1986.

[22] S. Wolf and R. N. Tauber, *Silicon Processing for the VLSI Era*, Sunset Beach: Lattice Press, 1987.

[23] R. R. Hocking, "The Analysis and Selection of Variables in Linear Regression," *Biometrics*, vol. 32, March, 1976.

[24] S. F. Lee, *Private Communication*, March, 1991.

[25] M. J. D. Powell, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculation," *Proceedings of the Dundee Conference on Numerical Analysis*, June, 1977.

[26] D. L. Flamm, V. M. Donnelly, and D. E. Ibbotson, "Basic Chemistry and Mechanisms of Plasma Etching," *Journal of Vacuum Science & Technology B*, vol. 1, no. 1, Jan-Mar, 1983.

[27] S. E. Bernacki and B. B. Kosicki, "Controlled Film Formation During CCl_4 Plasma Etching," *Journal of the Electrochemical Society*, vol. 131, no. 8, August, 1984.

APPENDIX 4.1

AN EMPIRICAL MODEL FOR ETCH TEMPERATURE

Motivation

The ultimate goal of the development of models which describe the behavior of the plasma etcher is to use these models for equipment diagnosis. Although the models for etch rate, uniformity, selectivity, and anisotropy provide accurate diagnostic information, they are only useful during the in-line measurement phase, after the actual etch process has been completed. A model describing a real-time process condition would provide additional diagnostic evidence which could be used to infer equipment malfunctions even more rapidly. Since the temperature of the Lam etcher process chamber is not directly controlled by the user [1], a model for this response could serve to capture the "signature" of the proper behavior of the machine.

Background

Process temperature has a profound influence on plasma discharge chemistry. The temperature of the gas mixture is a complex function of power input, heat transfer, and transport phenomena. The rate constants for the chemical reactions in the etching process vary with temperature according to the Arrhenius expression [2]:

$$k(T) = A(T)e^{-\frac{E_A}{RT}} \qquad \text{(a4.1.1)}$$

where A is a "pre-exponential" which is weakly dependent on temperature, and E_A is the "activation energy." Since the rate constants for chemical reactions are a function of temperature, the temperature also has an indirect effect on selectivity, uniformity and etch rate.

Metrology

In the Lam Autoetch 490 plasma etcher, the process temperature is controlled by means of a system which removes heat from the lower electrode by circulating deionized water [1]. However, even though this control scheme prevents the electrode temperature from elevating much above room temperature, subtle increases in temperature during processing are still detectable by the equipment monitoring system [3]. A typical graph depicting this phenomenon is shown in Figure a4.1.1

Since the temperature increases almost linearly with time, the temperature gradient may be described by the slope of the least squares regression line [4] of the temperature versus time plot. This slope is a function of the etch recipe. Thus, it is possible to model the process temperature gradient using response surface methods in the same manner as the in-line responses (i.e. - etch rate, uniformity, selectivity, etc.). Such a model has been developed using temperature data from the same two-phase experiment described in Chapter 4.

Temperature Gradient Model

Experimental data was analyzed by piecewise regression [5] using *R/S Discover* [6]. The regression model for the temperature gradient (∇T) is:

$$\nabla T = 7.29 - 0.006P + 0.036Rf + 0.03CCl_4 - 0.014He - 11.9G \qquad \text{(a4.1.2)}$$
$$+ (1.12e{-}4)P^2 - (2e{-}4)P{*}He + (9.85e{-}5)He^2 + 3.58G^2 \quad \pm 0.39\ (mdeg/sec)$$

The prediction error of the uniformity model is ± 0.39 millidegrees/second. The equipment replication error is ± 1.17 mdeg/sec. Significance tests reveal that all model coefficients are relevant, and there is no evidence for lack of fit. A scatterplot of the predicted gradient versus is shown in Figure a4.1.2. Figures a4.1.3 and a4.1.4 describe the behavior of the temperature gradient under various process conditions. Ideally, the gradient should be kept as low as possible to

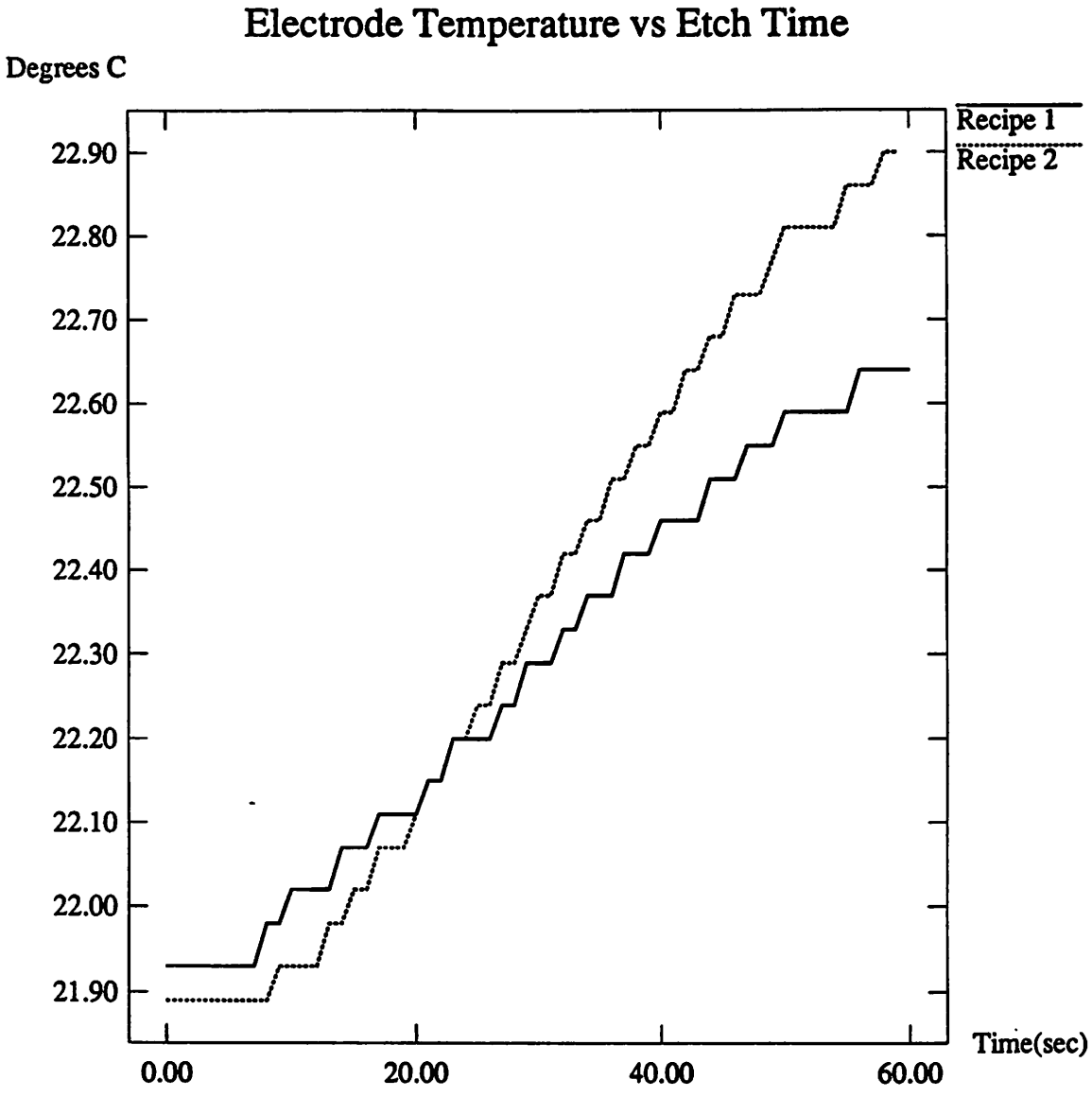


Figure a4.1.1 - Typical plot of process temperature versus time.

reduce the possibility of overheating and ensure the stability of long etching i steps. From these figures, it is evident that the a low gradient is achieved at low power and pressure as well as using wide electrode spacing. In addition, the presence of helium in the process chamber seems to serve as a cooling agent. The ANOVA table of this model appears below.

Table a4.1.1: ANOVA for Temperature Gradient Model

Source	DF	Sum of Squares	Mean Square	F-Ratio	Significance
Total	52	363.23	6.99		
Regression	9	304.54	33.84	24.80	0.000
Residual	43	58.68	1.37		
Lack of Fit	35	41.50	1.19	0.55	0.892
Error	8	17.19	2.15		

Adjusted $R^2 = 0.805$

Summary

This appendix presents an empirical model for the temperature gradient in the plasma etch process chamber. This model possesses the advantage of providing additional diagnostic information during the real-time phase of the process. Although this model has yet to be incorporated into the overall process control and diagnostic strategy for the etcher, its use in conjunction with the equipment monitoring scheme [3] will potentially allow the detection of failures early in the fabrication cycle.

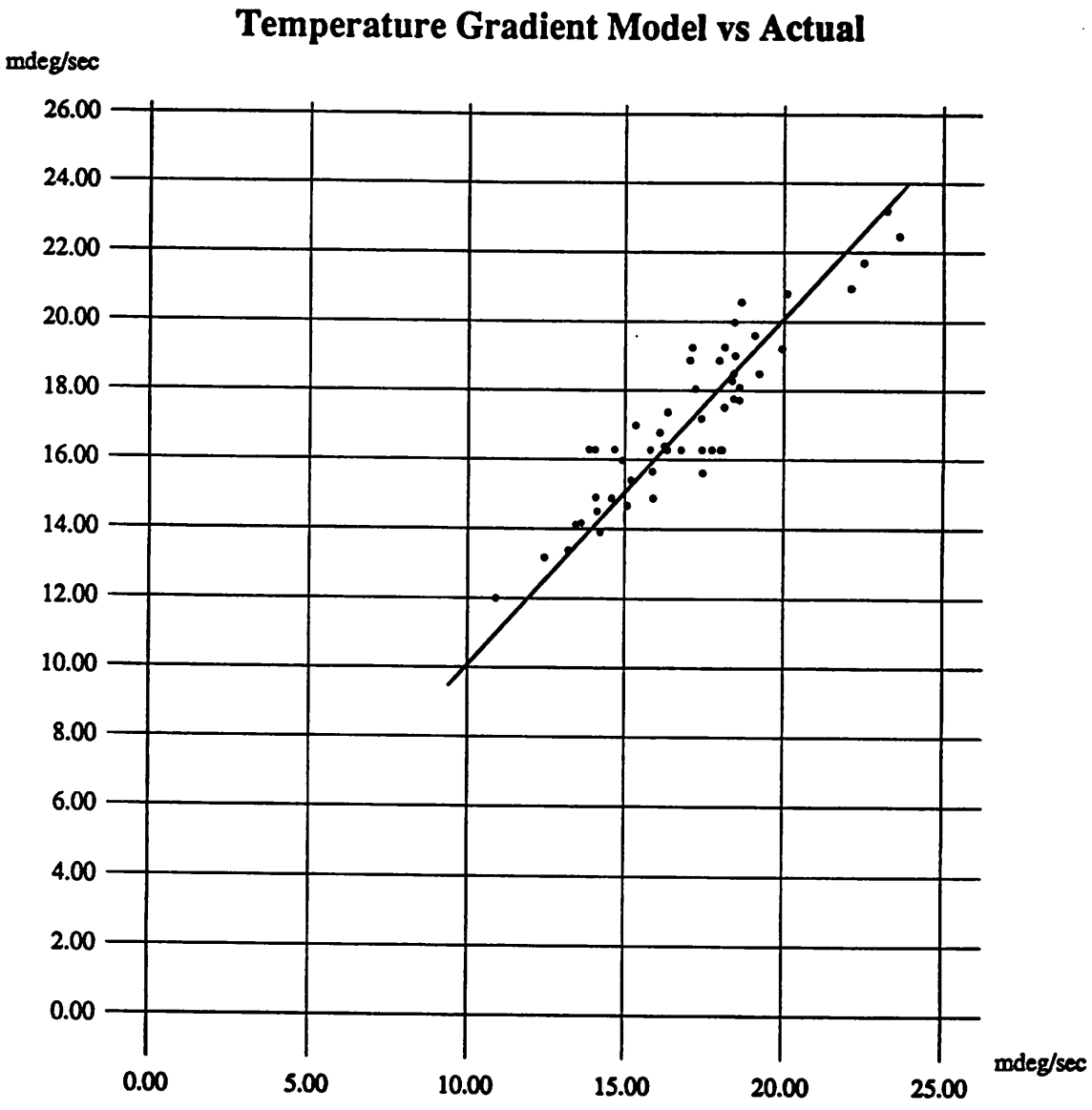


Figure a4.1.2 - Scatterplot of predicted temperature gradients vs actual measured gradients.

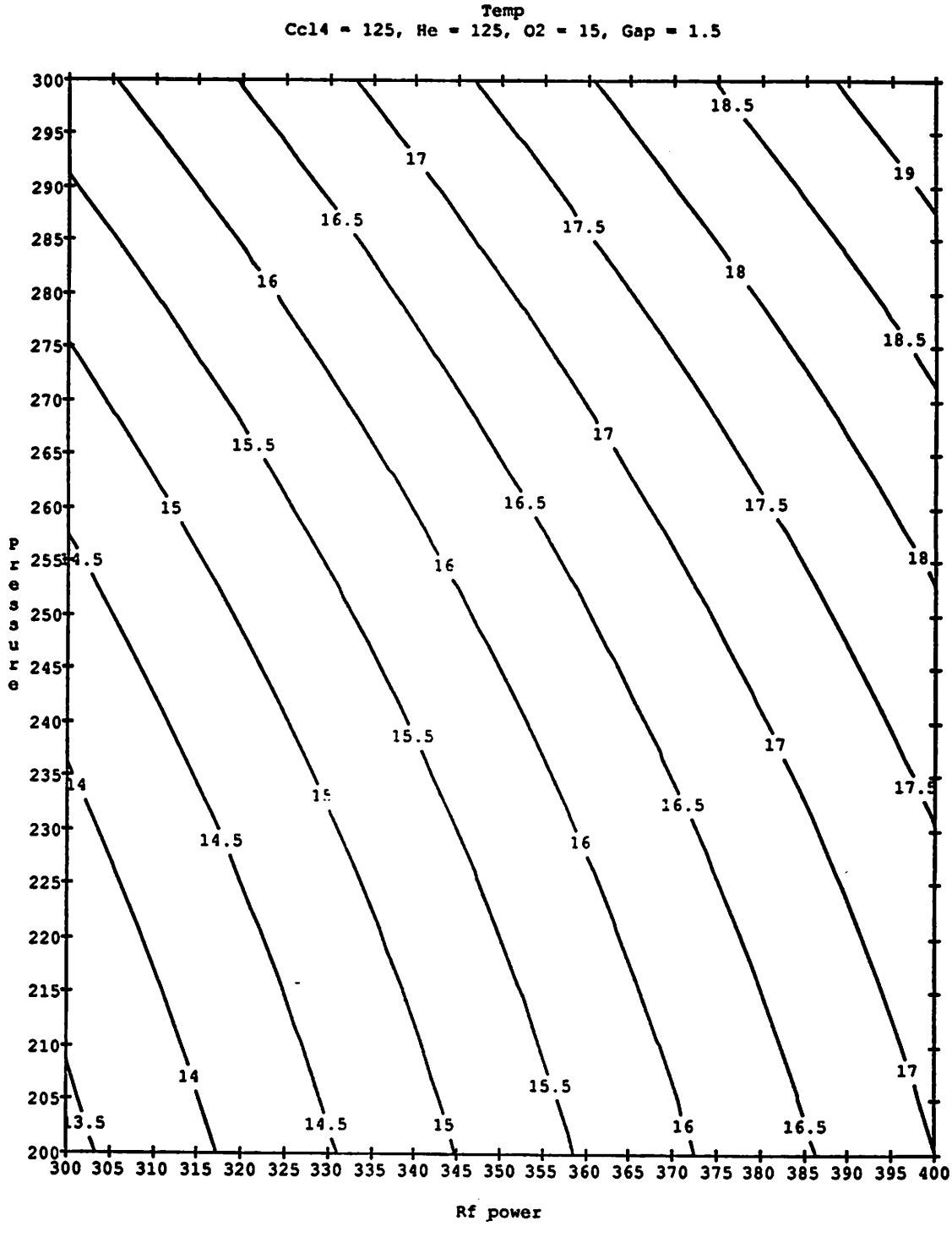


Figure a4.1.3 - Contour plot of the temperature gradient versus RF power and pressure.

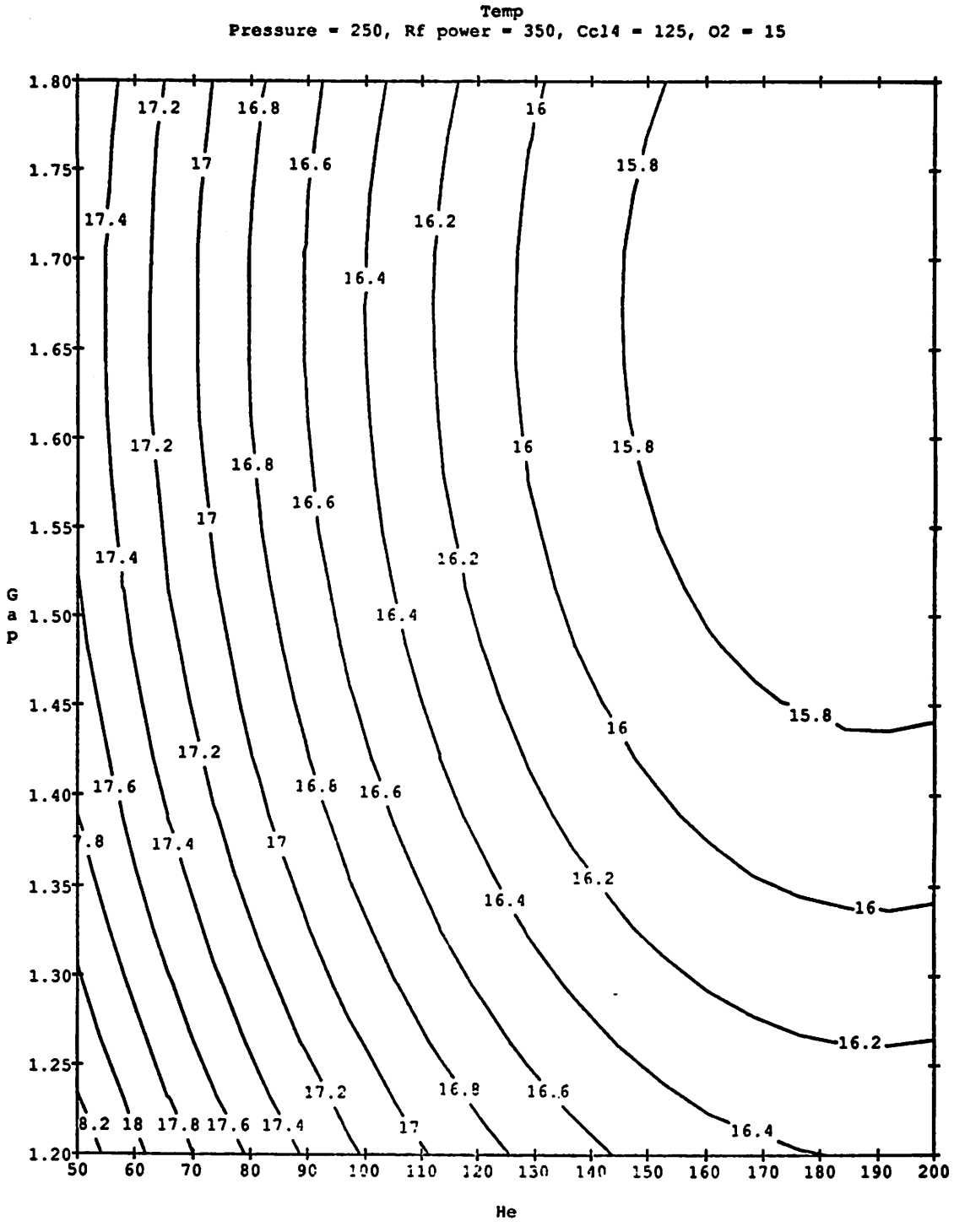


Figure a4.1.4 - Contour plot of the temperature gradient versus electrode spacing and *He* flow.

References for Appendix 4.1

[1] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, March, 1985.

[2] D. M. Manos and D. L. Flamm, *Plasma Etching: An Introduction*, San Diego: Academic Press, 1989.

[3] N. H. Chang, A. Gaduh, H. Guo, and D. Mudie, "A Real-Time Equipment Monitoring Program," *Electronics Research Laboratory Research Summary*, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1990.

[4] G. E. P. Box, W. B. Hunter, and J. S. Hunter, *Statistics for Experimenters*, New York: Wiley, 1978.

[5] R. R. Hocking, "The Analysis and Selection of Variables in Linear Regression," *Biometrics*, vol. 32, March, 1976.

[6] *RS/Discover User's Guide*, BBN Software Products Corporation, June 1988.

APPENDIX 4.2

AN OBJECT-ORIENTED SOFTWARE LIBRARY FOR BCAM EQUIPMENT MODELS

Motivation

The characterization of integrated circuit processes through equipment models such as those developed in Chapter 4 has become a necessity in semiconductor manufacturing. Equipment models may be physical or empirical, or a combination of both. Further, equipment-specific models are often updated to reflect the changing status of the equipment over time. In addition to the models for the Lam Research Automated Plasma Etcher [1], the Berkeley Computer-Aided Manufacturing (BCAM) group has also developed several other statistically-based polynomial models to describe the behavior of important pieces of IC manufacturing equipment, including the Tylan Low-Pressure Chemical Vapor Deposition (LPCVD), and the photolithography workcell [2,3].

In order to aid in the ongoing derivation of further equipment models, the BCAM group has also developed an object-oriented model library. The overall purpose of the library is to provide an efficient means of storing, retrieving, evaluating, analyzing, and otherwise manipulating these models for use by other modules in the BCAM architecture (i.e. - recipe generation, malfunction diagnosis, statistical process control, etc.). Further, the library makes the models available to simulation tools such as the SIMPL-DIX TCAD software package. The entire library is written in C++, an object-oriented superset of the C programming language [4].

Rationale for Object-Oriented Approach

The BCAM system currently employs equipment models for the Lam Research automated plasma etcher, the Tylan low-pressure chemical vapor deposition furnace, the Eaton photoresist

coating and baking system, the GCA wafer stepper, and the MTI development station. In addition, it is anticipated that many more models may be developed in the future. Given its inherent modularity, hierarchical nature, and inheritance properties, new models may be added in an object-oriented environment with relative ease [5]. Consequently, it was determined that the BCAM equipment model library should be implemented using an object-oriented programming (OOP) language. Due to the overall efficiency of the C programming language and its ability to interface with other software in the UNIX programming environment, the language chosen for this application was C++, an object-oriented extension of C. (For a more detailed description of the advantages of OOP and C++, refer to Chapter 6).

Model Library Structure

The class structure of the equipment model library is depicted in Figure a4.2.1. In this structure, the most general object class is the generic *model*. All subsequent classes inherit their basic data structures and methods from the *model* class. There are two types of models derived from this parent class: *process* models and *equipment* models. *Process* models are "textbook" physical models of fabrication processes such as the well-known Deal-Grove model for silicon oxidation [6]. The *equipment* model class, on the other hand, contains the empirical and semi-empirical equipment-specific models for the Lam etcher, Tylan furnace, and other fabrication equipment located in the Berkeley Microlab. Each *equipment* class possesses several subclasses which consist of the specific characteristics modeled in that particular piece of equipment. For the etcher, these subclasses include: *etch rate*, *uniformity*, *selectivity*, and *anisotropy*. These models are discussed in detail in Chapter 4.

The C++ code which implements the structure of the generic *model* class and all of its subclasses appears in Appendix 4.2a. The parent class contains the following crucial pieces of data:

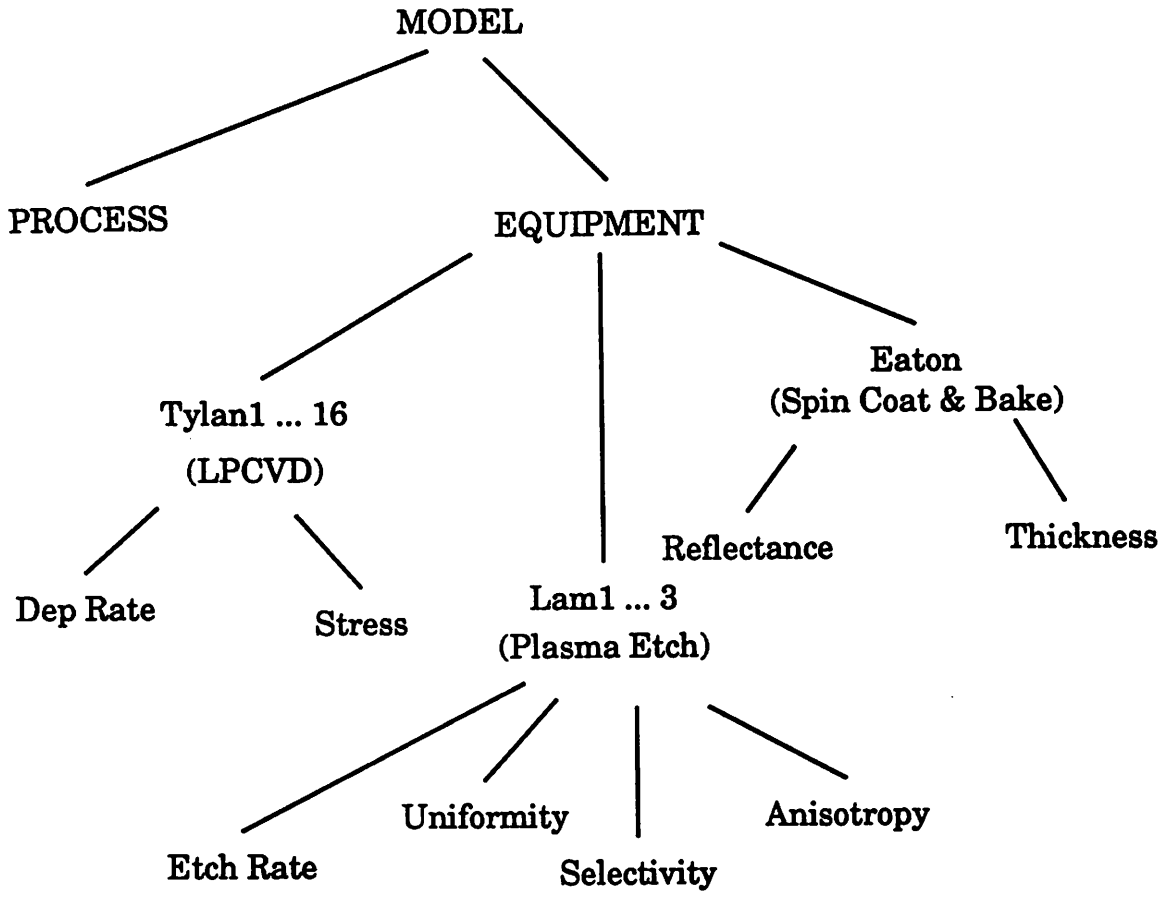


Figure a4.2.1 - Hierarchical description of BCAM model objects.

- *version* - a character string which denotes the version designation of the particular model.
- *term_cnt* - the total number of terms in the model.
- *param_cnt* - the total number of independent parameters in the model.
- *coeffs* - the array which contains the coefficients of the model terms.
- *param* - the array which contains the actual parameter values for a given recipe.
- *exp_mat* - the matrix of dimension [*term_cnt,param_cnt*] which contains the exponents to which the parameters in each model term are raised.

- *std_err* - the standard error (or equivalently, "replication error") of the equipment that is being modeled.
- *pred_err* - the prediction error of the model.
- *param_names* - a string array containing the names of the model parameters.

In order to more clearly illustrate the meaning of the data structures in the *model* class, consider the model derived for polysilicon etch rate (R_p) in the Lam plasma etcher:

$$\begin{aligned} R_p = &-245 - 4.24P + 11.0Rf + 0.742CCl_4 + 11.2He \\ &+ 523G + 35.9O_2 - 0.034P*He + 7.82P*G \\ &+ 0.085Rf*CCl_4 - 8.36Rf*G - 0.132(CCl_4)^2 \\ &+ 0.059CCl_4*He - 0.059He^2 \end{aligned} \tag{a4.2.1}$$

For this particular model, the above variables would assume the following values:

$$version = 1.0 \tag{a4.2.2}$$

$$term\_cnt = 14 \tag{a4.2.3}$$

$$param\_cnt = 6 \tag{a4.2.4}$$

$$coeffs = \begin{bmatrix} -245 \\ -4.24 \\ 11.0 \\ 0.742 \\ 11.2 \\ 523 \\ 35.9 \\ -0.034 \\ 7.82 \\ 0.085 \\ -8.36 \\ -0.132 \\ 0.059 \\ -0.059 \end{bmatrix} \tag{a4.2.5}$$

$$param = \begin{bmatrix} recipe\ pressure\ value \\ recipe\ power\ value \\ recipe\ CCl_4\ value \\ recipe\ He\ value \\ recipe\ O_2\ value \\ recipe\ gap\ value \end{bmatrix} \tag{a4.2.6}$$

$$exp\_mat = \begin{bmatrix} P & Rf & CCl_4 & He & O_2 & G \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix} \tag{a4.2.7}$$

$$std\_err = 309 \tag{a4.2.8}$$

$$pred\_err = 104 \tag{a4.2.9}$$

$$param\_names = \begin{bmatrix} P \\ Rf \\ CCl_4 \\ He \\ O_2 \\ G \end{bmatrix} \tag{a4.2.10}$$

Functionality of Model Objects

The primary methods which perform operations on the basic *model* object in Appendix 4.2a are as follows:

- *eval* - this function evaluates a particular model response for a given recipe of parameters.

- *get_std_err* - this function returns the standard error of the model.
- *get_pred_err* - this function returns the prediction error of the model.
- *sens* - this function evaluates the sensitivity of the model at a given point in the parameter space. Sensitivity is defined as the first partial derivative of the polynomial). The derivatives are evaluated analytically for polynomial models and numerically for any other model type.
- *get_param_index* - this function returns the index of a given parameter in the *param_names* array.
- *print_model* - this function prints models in an aesthetically pleasing format.
- *update_model* - this function is intended to provide a means of updating a particular version of a model by adjusting its coefficients. It has yet to be written.
- *store_version* - this function stores the model along with its version number as a character string in the INGRES database [7]. It has yet to be written.
- *find_version* - this function retrieves the character string representation of a model from the INGRES database using its version number. It has yet to be written. The function is accompanied by a routine which parses the character string (separating terms, counting terms, counting parameters, identifying exponents, identifying coefficients, etc.) so that the model may be manipulated by other methods such as *sens* and *eval*. The parsing routine is fully operational.
- *delete_version* - this function deletes a particular version of a model from the database. It has yet to be written.

These functions assume that all models are polynomials expressions. However, special methods have been adapted to models where this is not the case (such as the LPCVD deposition rate

model [3]). Using "virtual" C++ functions [4], such adaptation is relatively straightforward.

Summary and Future Work

The basic structure and necessary functions of the BCAM equipment model library have been designed, implemented and tested. However, several methods currently require further development. Most of these, such as the *store_version*, *find_version*, and *delete_version* functions, merely involve the connection between the library and the INGRES database. In addition, a robust method for updating model coefficients and terms must also be designed.

Models for the Lam etcher and Tylan furnace are presently accessible from the library. Plans are underway for the inclusion of the models for the components of the photolithography workcell (i.e. - the Eaton coating and baking system, GCA wafer stepper, and MTI developer) as soon as the development of these models is complete [8]. Existing models are currently being used for recipe generation, malfunction diagnosis, and technology CAD (TCAD) development. C++ code for the library resides on 'radon.berkeley.edu' in the ~bcam/src/models directory. The code which implements generic model functions as well as the Lam etcher model is given in Appendices 4.2b and 4.2c, respectively.

References for Appendix 4.2

[1] G. S. May, J. Huang, and C. J. Spanos, "Experimental Modeling of the Etch Characteristics of Polysilicon in $CCl_4/He/O_2$ Plasmas," *Proceedings of the 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium*, October, 1990.

[2] K. K. Lin, and C. J. Spanos, "Statistical Equipment Modeling for VLSI Manufacturing: An Application for LPCVD," *IEEE Transactions on Semiconductor Manufacturing*, vol. 3, no. 4, November, 1990.

[5] Z. M. Ling and C. J. Spanos, "In-line Supervisory Control in a Photolithographic Workcell," *Proceedings of the Fifth Annual SRC/DARPA CIM-IC Workshop*, August, 1990.

[4] K. Weiskamp and B. Flaming, *The Complete C++ Primer*, San Diego: Academic Press, 1989.

[5] J. Besemer, "An Introduction to Object-Oriented Design and Programming," *CASE Outlook*, 1987.

[6] B. E. Deal and A. S. Grove, "General Relationship for the Thermal Oxidation of Silicon," *Journal of Applied Physics*, vol. 36, p. 3770, 1965.

[7] *INGRES/Embedded SQL Companion Guide for C*, Relational Technologies, Inc., 1989.

[8] S. Leang, *Private Communication*, February, 1991.

APPENDIX 4.2a

C++ IMPLEMENTATION OF MODEL LIBRARY CLASS STRUCTURE

```
/*
```

Model definition module of BCAM

Author: *gsm*
Source: */home/radon 1/bcam/src/models*
Revision: 1.0
Date: 90/02/01 09:09:59

FileName: *model.h*

```
*/

//
// Symbolic constants for BCAM models.
//

#define    MAXCOEFF    50    // maximum number of model coefficients
#define    MAXPARAM    10    // maximum number of model parameters

//*****************************************************************************


//
// Class declarations for all BCAM models and their associated methods.
//


// Basic model class

class model {
```

```
protected:
        char *version;
        int term_cnt, param_cnt;
        long exp_mat[50][10];
        double std_err, pred_err;
        double param[10], coeffs[50];
        char *param_names[10];
public:
        virtual double eval(double param[]);
        double get_std_err(float n = 1.0) { return n*std_err; }
        double get_pred_err(float n = 1.0) { return n*pred_err; }
        virtual double sens(int n, double param[]);
        int get_param_index(char *name);
        void print_model();
        void update_model();
        friend void store_version();
        friend model find_version();
        friend void delete_version();
};


// ********************************************************************************

// Process model class

class process : public model {
        char *process_name;
        void print_proc_name();
};


// Equipment model class

class equipment : public model {
        char *equip_spec;
        void print_equip_spec();
        void update_equip_spec();
};


// ********************************************************************************

// Lam1 model class

class lam1 : public equipment {
        int response_cnt;
        char *response_names[10];
        void update_response_names();
public:
```

```
        void std_lam1_model(void);
};

// Tylan16 model class

class tylan16 : public equipment {
        int response_cnt;
        char *response_names[10];
        void update_response_names();
public:
};

// ***************************************************************************

// Lam1 etch rate model

class etch_rate : public lam1 {
public:
        void std_etch_rate_model(void);
};

// Lam1 etch unifomity model

class uniformity : public lam1 {
public:
        void std_uniformity_model(void);
};

// Lam1 oxide selectivity model

class sox : public lam1 {
public:
        void std_sox_model(void);
};

// Lam1 resist selectivity model

class sph : public lam1 {
public:
        void std_sph_model(void);
};

// Lam1 etch anisotropy model
```

```
class anisotropy : public lam1 {
public:
        void std_anisotropy_model(void);
};

// Lam1 temperature gradient model

class temp_grad : public lam1 {
public:
        void std_temp_grad_model(void);
};


// Tylan16 deposition rate model

class dep_rate : public tylan16 {
public:
        double eval(double param[]);
        double sens(int n, double param[]);
        void std_dep_rate_model(void);
};


// Tylan16 stress model

class stress : public tylan16 {
public:
        void std_stress_model(void);
};
```

APPENDIX 4.2b

C++ IMPLEMENTATION OF MODEL LIBRARY GENERIC METHODS

```
/*
```

Model definition module of BCAM

Author: *gsm*
Source: */home/radon 1/bcam/src/models*
Revision: 1.0
Date: 90/02/01 09:09:59

FileName: *model.cc*

```
*/

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


// ****************************************************************************

//
// Member function definitions for model class.
//


// This function returns the value of the response model for a previously set
// array of parameter values ("param[]").

double model::eval(double param[])
{
        int i,j;
        double sum, term;
```

```
        sum = 0.0;
        for (i = 0; i<term_cnt; i++) {
                term = 1.0;
                for (j = 0; j<param_cnt; j++) {
                        term *= pow(param[j], exp_mat[i][j]);
                }
                sum += coeffs[i] * term;
        }
        return sum;
}


// Return the first partial derivative of the model with respect to parameter
// "n" and evaluated at the array of parameter values "param[]".

double model::sens(int n, double param[])
{
        int i,j;
        long new_exp_mat[MAXCOEFF][MAXPARAM];
        double sum, term;
        double new_coeffs[MAXCOEFF];

        for (i = 0; i<term_cnt; i++) {
                for (j = 0; j<param_cnt; j++) {
                        new_exp_mat[i][j] = exp_mat[i][j];
                }
                if (exp_mat[i][n] != 0) {
                        new_coeffs[i] = coeffs[i] * exp_mat[i][n];
                        new_exp_mat[i][n] -= 1;
                } else {
                        new_coeffs[i] = 0.0;
                }
        }

        sum = 0.0;
        for (i = 0; i<term_cnt; i++) {
                term = 1.0;
                for (j = 0; j<param_cnt; j++) {
                        term *= pow(param[j], new_exp_mat[i][j]);
                }
                sum += new_coeffs[i] * term;
        }
        return sum;
}


// Return the index of the "param_names" array corresponding to the given
// parameter "name".
```

```
int model::get_param_index(char *name)
{
        int i;
        for (i = 0; i<param_cnt; i++) {
                if (param_names[i] = name) return i;
        }
}
```

APPENDIX 4.2c

C++ IMPLEMENTATION OF METHODS FOR LAM ETCHER MODELS

```
/*

  Model definition module of BCAM

  Copyright (c) 1990 Regents of the University of California

      Permission to use, copy, modify, and distribute this software and its documentation for any
      purpose and without fee is hereby granted, provided that the above copyright notice appear
      in all copies and that both that copyright notice and this permission notice appear in sup-
      porting documentation, and that the name of the University of California not be used in
      advertising or publicity pertaining to distribution of the software without specific, written
      prior permission. The University of California makes no representations about the suitabil-
      ity of this software for any purpose. It is provided "as is" without express or implied war-
      ranty.

  Author: gsm
  Source: /home/radon 1/bcam/src/models
  Revision: 1.0
  Date: 90/02/01 09:09:59

  FileName: lam 1.cc

*/

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


// ***************************************************************************

//
// Member functions of the lam1 class.
//

// Set up the standard lam1 model.

void lam1::std_lam1_model(void)
{
      int i,j;

      term_cnt = 28;
```

```
param_cnt = 6;

param_names[0] = "P";
param_names[1] = "Rf";
param_names[2] = "CCl4";
param_names[3] = "He";
param_names[4] = "O2";
param_names[5] = "G";


// Initialize and set exponent matrix.

for (i = 0; i<term_cnt; i++) {
        for (j = 0; j<param_cnt; j++) {
                exp_mat[i][j] = 0;
        }
}
exp_mat[1][0] = 1;
exp_mat[2][1] = 1;
exp_mat[3][2] = 1;
exp_mat[4][3] = 1;
exp_mat[5][4] = 1;
exp_mat[6][5] = 1;
exp_mat[7][0] = 2;
exp_mat[8][0] = 1;
exp_mat[8][1] = 1;
exp_mat[9][0] = 1;
exp_mat[9][2] = 1;
exp_mat[10][0] = 1;
exp_mat[10][3] = 1;
exp_mat[11][0] = 1;
exp_mat[11][4] = 1;
exp_mat[12][0] = 1;
exp_mat[12][5] = 1;
exp_mat[13][1] = 2;
exp_mat[14][1] = 1;
exp_mat[14][2] = 1;
exp_mat[15][1] = 1;
exp_mat[15][3] = 1;
exp_mat[16][1] = 1;
exp_mat[16][4] = 1;
exp_mat[17][1] = 1;
exp_mat[17][5] = 1;
exp_mat[18][2] = 2;
exp_mat[19][2] = 1;
exp_mat[19][3] = 1;
exp_mat[20][2] = 1;
exp_mat[20][4] = 1;
exp_mat[21][2] = 1;
```

```
        exp_mat[21][5] = 1;
        exp_mat[22][3] = 2;
        exp_mat[23][3] = 1;
        exp_mat[23][4] = 1;
        exp_mat[24][3] = 1;
        exp_mat[24][5] = 1;
        exp_mat[25][4] = 2;
        exp_mat[26][4] = 1;
        exp_mat[26][5] = 1;
        exp_mat[27][5] = 2;
}

//*****************************************************************************

//
// Member function definitions for etch rate class.
//

// Set up the standard etch_rate model.

void etch_rate::std_etch_rate_model(void)
{
        int i;

        std_err = 309.4;
        pred_err = 104.0;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = -244.954;
        coeffs[1] = -4.239;
        coeffs[2] = 10.96;
        coeffs[3] = 0.742;
        coeffs[4] = 11.225;
        coeffs[5] = 35.868;
        coeffs[6] = 523.149;
        coeffs[10] = -0.034;
        coeffs[12] = 7.817;
        coeffs[14] = 0.085;
        coeffs[17] = -8.362;
        coeffs[18] = -0.132;
        coeffs[19] = 0.059;
        coeffs[22] = -0.059;
}
```

```
//
// Member function definitions for uniformity class.
//

// Set up the standard uniformity model.

void uniformity::std_uniformity_model(void)
{
        int i;

        std_err = 6.66;
        pred_err = 2.22;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = -11.549;
        coeffs[1] = -0.038527;
        coeffs[2] = 0.093699;
        coeffs[3] = 0.709912;
        coeffs[4] = -0.415408;
        coeffs[6] = -8.898964;
        coeffs[14] = -0.00177;
        coeffs[15] = 0.001383;
        coeffs[19] = -0.001403;
        coeffs[22] = 7.975e-4;
}

//
// Member function definitions for oxide selectivity class.
//

// Set up the standard oxide selectivity model.

void sox::std_sox_model(void)
{
        int i;

        std_err = 0.93;
        pred_err = 0.31;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = -13.106;
```

```
        coeffs[1] = 0.097;
        coeffs[2] = 0.04;
        coeffs[3] = -0.06;
        coeffs[4] = 0.059;
        coeffs[5] = 0.079;
        coeffs[8] = -2e-4;
        coeffs[9] = 2.898e-4;
        coeffs[10] = -3e-4;
}

//
// Member function definitions for resist selectivity class.
//

// Set up the standard resist selectivity model.

void sph::std_sph_model(void)
{
        int i;

        std_err = 0.95;
        pred_err = 0.09;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = 7.558;
        coeffs[1] = 0.009;
        coeffs[2] = -0.014;
        coeffs[3] = -0.022;
        coeffs[4] = 0.006;
        coeffs[5] = -0.099;
        coeffs[6] = -2.586;
        coeffs[8] = -5e-5;
        coeffs[9] = 1.292e-4;
        coeffs[10] = -7e-5;
        coeffs[11] = 3.688e-4;
        coeffs[13] = 2.713e-5;
        coeffs[15] = 3.558e-5;
        coeffs[19] = -5e-5;
        coeffs[27] = 0.757;
}

//
// Member function definitions for anisotropy class.
//
```

```
// Set up the standard anisotropy model.

void anisotropy::std_anisotropy_model(void)
{
        int i;

        std_err = 3.24;
        pred_err = 1.82;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = 94.61;
        coeffs[4] = -0.031;
}

//
// Member function definitions for temperature gradient class.
//

// Set up the standard temperature gradient model.

void temp_grad::std_temp_grad_model(void)
{
        int i;

        std_err = 1.17;
        pred_err = 0.39;

// Initialize and set coefficient array.

        for (i = 0; i<term_cnt; i++) {
                coeffs[i] = 0.0;
        }
        coeffs[0] = 7.287;
        coeffs[1] = -0.006;
        coeffs[2] = 0.036;
        coeffs[3] = 0.03;
        coeffs[4] = 0.014;
        coeffs[6] = -11.946;
        coeffs[7] = 1.116e-4;
        coeffs[10] = -2e-4;
        coeffs[22] = 9.846e-5;
        coeffs[27] = 3.579;
}
```

CHAPTER 5

GENERATION AND DISTRIBUTION OF EVIDENTIAL SUPPORT

5.1 Introduction

The previous two chapters have discussed the mechanical operation of a plasma etcher and the development of quantitative models for etch behavior. In Dempster-Shafer (D-S) terminology, these two chapters have established the *frame of discernment* and *governing equations* for equipment diagnosis [1-2]. The only remaining task necessary to fully implement the Dempster-Shafer approach to fault diagnosis is the development of a set of techniques to generate and distribute numerical evidential support among all of the various fault hypotheses in the etcher's frame of discernment. The objective of this chapter is to provide such techniques, and in so doing, complete the formulation of the methodology for IC equipment malfunction diagnosis.

5.2 Chronology of Evidence Availability

Due to the batch nature of the semiconductor manufacturing process, useful diagnostic information, or *evidence*, accumulates over time in an irregular fashion. Several sources of evidence are available, but all of this information may be placed into one of three major categories according to the chronology in which it is obtained. Evidence regarding equipment status may be collected either: 1) during *maintenance* periods (before processing), 2) during *on-line* equipment operation (during processing), or 3) during *in-line* physical and/or electrical inspection of the processed wafer (after processing) [3-5]. These chronological evidence sources are illustrated in Figure 5.1.

Using these three categories of evidence as a framework, malfunction diagnosis takes place in three consecutive steps. *Maintenance diagnosis* is performed by examining the relevant historical records of equipment performance. Next, *on-line diagnosis* is performed based on analysis

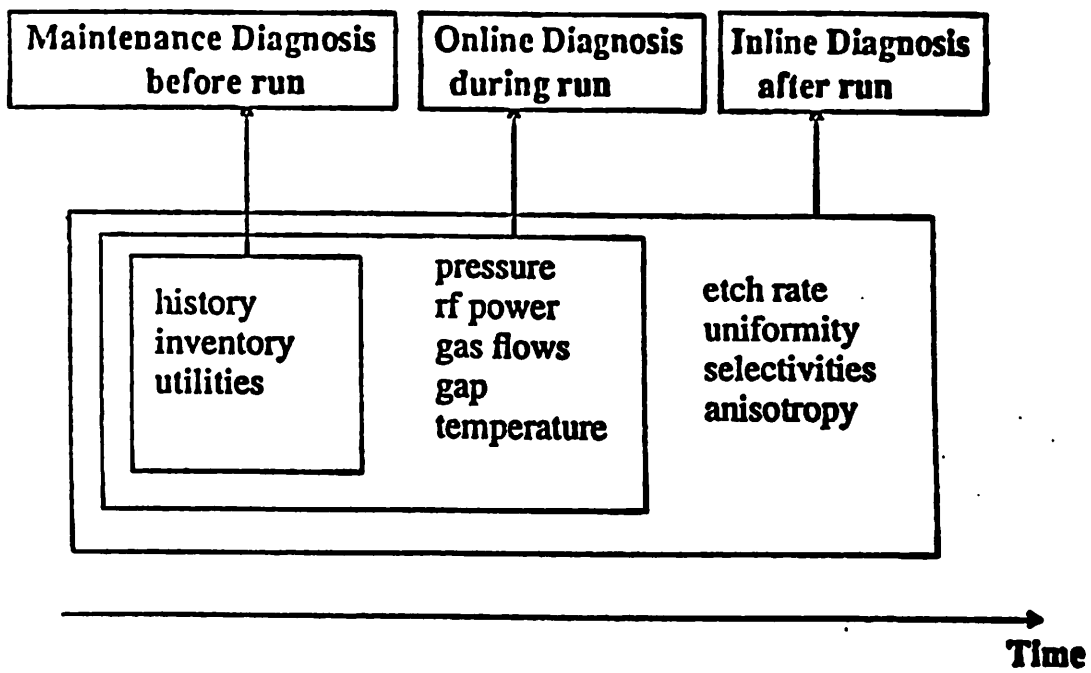


Figure 5.1 - Chronological sources of diagnostic evidence.

of real-time sensor data available from equipment monitoring facilities. Finally, *in-line diagnosis* makes use of physical and/or electrical in-line measurements on processed wafers in conjunction with empirical models for equipment behavior. For each of these three phases of diagnosis, evidential support for fault hypotheses is generated and mapped to particular malfunctioning equipment components. The methodology employed to generate support and obtain this mapping is discussed below.

5.3 Generation of Evidential Support

5.3.1 The Support Function

The Dempster-Shafer evidence combination method illustrated in Chapter 2 (see § 2.4) depends heavily upon the basic manner in which evidential support is initially distributed to the

BPMDs. In a Boolean classification scheme, the belief in any fault hypothesis is set equal to one as soon as the measurement residual corresponding to a particular constraint exceeds its absolute tolerance (*tol*). This residual is defined as the difference between a measured response and its setpoint. This type of belief (or *support*) assignment is depicted by the step function in Figure 5.2. However, this step function yields highly unstable belief assignment in the presence of measurement noise [2-5]. Thus, incremental changes in equipment operating conditions when constraints are at or near *tol* lead to discontinuous adjustments in the support values associated with various diagnoses.

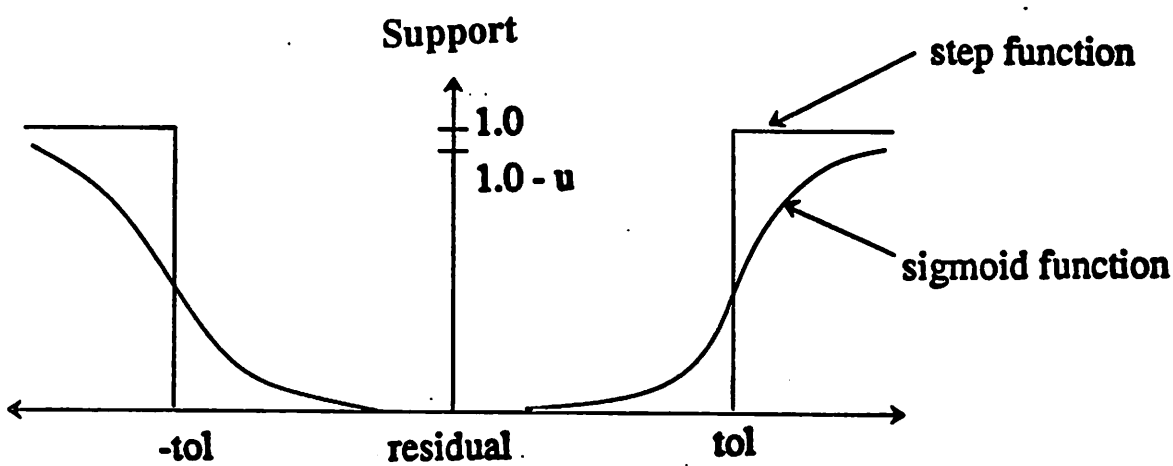


Figure 5.2 - Boolean and sigmoid support functions [3-5].

A better behaving support function is the sigmoid function also shown in Figure 5.2. This "squashing" function has been used in neural network studies [6]. The equation for the sigmoid function is:

$$s(\varepsilon) = \frac{1-u}{1+\exp[-G(\frac{\varepsilon}{tol}-1)]} \tag{5.1}$$

where u is the uncertainty associated with the individual constraint and ε is the measurement residual, which is assumed to be a normally distributed random variable with mean zero and constant variance (σ^2). The parameter G is an arbitrary constant used to adjust the sharpness of the function. As G approaches infinity, the sigmoid function approximates the shape of the step function.

Values for the adjustable parameters u and G are typically based on experience. However, the default value of G is set at 5.0, which causes the support for a given hypothesis to be less than 0.1 when ε is 2-σ and larger than 0.2 when ε is 2.5-σ (assuming zero uncertainty) [3-4]. The default value of uncertainty in the system is 0.1. As indicated below, this support function forms the basis for the generation of numerical belief in two of the three phases of malfunction diagnosis.

5.3.2 Support Generation for Maintenance Diagnosis

Computerized equipment maintenance records allow the maintenance phase of diagnosis to take place in a straightforward manner. Two quantities are used to characterize the likelihood that a particular component is faulty: 1) its mean-time-between-failures (*mtf*), and 2) the elapsed time since its last failure (*tsf*). These quantities may be extracted directly from FAULTS, UC-Berkeley's automated equipment maintenance record-keeping system [7]. Given the repair history of a particular component, evidential support may be generated in a similar manner to (5.1) using:

$$s(tsf) = \frac{1-u}{1+\exp[-G(\frac{tsf}{mtf}-1)]} \tag{5.2}$$

Here the elapsed time since the last equipment failure is treated as the measurement residual and the mean-time-between failures assumes the role of the measurement tolerance.

As an example, suppose that a mass flow controller must be calibrated every six weeks and, at the time that diagnosis takes place, it has been five weeks since its last calibration. If $G = 5.0$ and $u = 0.1$ (10% uncertainty in the system), then 0.27 units of support are attributed to the mis-calibrated mass flow controller fault. During maintenance diagnosis, support values for all components in the frame of discernment are calculated in this way.

5.3.3 Support Generation for On-line Diagnosis

Data from real-time equipment sensors serves as the source of evidence which is analyzed during the on-line phase of diagnosis. In the case of the plasma etcher, this data is transmitted via a software system known as LAMTALK [8]. LAMTALK sends the sensor readings over a local area network to a workstation controlling the process. The program monitors 11 current process conditions which provide useful diagnostic information. These conditions include gas flow rates, forward and reflected RF power, various chamber and airlock pressures, chamber temperature, and electrode spacing (see § 4.2).

The real-time sensor data is sampled at a rate of one sample/second. At such a high sampling rate, it is reasonable to expect significant auto- and cross-correlation among the samples [9]. In order to alleviate these effects, a *CUSUM* control chart technique has been implemented to detect small process shifts [10]. The ability to detect small shifts is critical for the on-line diagnosis of rapid fabrication steps such as plasma etching. This is because slight equipment miscalibrations may only have sufficient time to manifest themselves as small shifts when the total processing time is on the order of minutes. CUSUM charts monitor such shifts by comparing the cumulative sums of the deviations of the sample values from their targets. This is accomplished by means of the *V-mask* shown in Figure 5.3. The procedure for detecting process shifts consists of determining whether or not all of the previous cumulative sums lie within the arms of the V-mask on the CUSUM control chart.

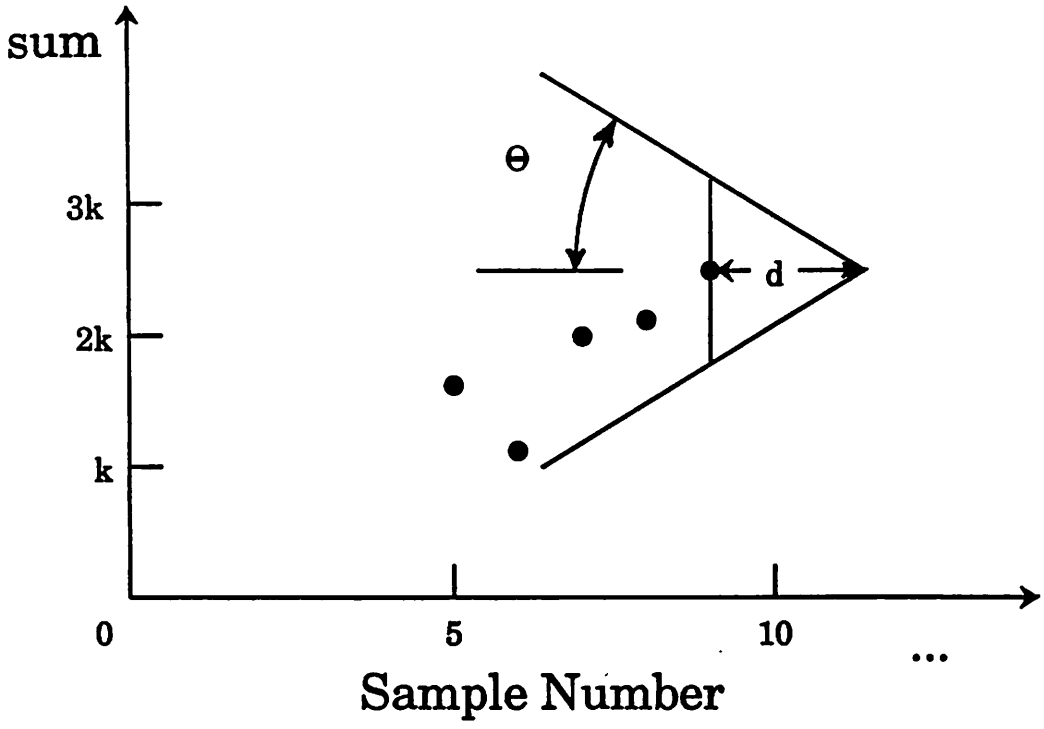


Figure 5.3 - Typical CUSUM control chart showing the V-mask and scaling parameters [10].

Using this approach to generate evidential support requires the following two cumulative sums:

$$S_H(i) = \max\,[0, \bar{x} - (\mu_0 + b) + S_H(i-1)] \tag{5.3}$$

$$S_L(i) = \max\,[0, (\mu_0 - b) - \bar{x} + S_H(i-1)] \tag{5.4}$$

where S_H is the sum used to detect positive process shifts, S_L is the sum used to detect negative shifts, $\bar{x}$ is the mean value of the current sample, and μ_0 is the target value. The initial value of both S_H and S_L is set to zero. Both sums accumulate deviations from the target value μ_0 greater than b, and both reset to zero upon becoming negative. The parameter b is given by:

$$b = tan(2\theta\sigma_x) \tag{5.5}$$

where σ_x is the standard deviation of the sampled variable and θ is the aspect angle of the V-

mask. This angle has been selected in such a way as to detect one-sigma process shifts with an average run length of 50 wafers between alarms when the process is in control.

When either S_L or S_H exceeds a decision interval h, this signals that the process has shifted out of statistical control. The decision interval is:

$$h = 2d\sigma_x \tan(\theta) \tag{5.6}$$

where d is the V-mask lead distance (see Figure 5.3). The decision interval may be used as the process tolerance limit and the sums S_H and S_L are be treated as measurement residuals [4]. Thus, numerical support is derived from the CUSUM chart using the sigmoid:

$$s(S_{H/L}) = \frac{1-u}{1+\exp[-G(\frac{S_{H/L}}{h}-1)]} \tag{5.7}$$

This function is used to generate Dempster-Shafer support values for all fault hypotheses related to the sensor data monitored during the on-line phase of diagnosis.

5.3.4 Plausibility Generation for In-line Diagnosis

In the final phase of diagnosis, in-line wafer measurements are used in conjunction with empirically obtained regression models of equipment behavior to obtain further numerical evidence. For the plasma etch application, the development of these models was described in Chapter 4. The responses that have been modeled are the etch rate, uniformity, oxide and photoresist selectivity, and anisotropy of a CCl_4-based polysilicon etch process. These responses are controlled by six input factors: pressure, RF power, electrode spacing, and the CCl_4, He, and O_2 flow rates.

For the purpose of generating numerical belief from the empirical models, it is assumed that a malfunction may be explained by an inadvertent shift in *one* of these six settings. Under this assumption, the system of regression equations is then used to test the individual hypotheses that

each of the settings has drifted from its target value. Statistical tests based on solving the system of regression equations in "reverse" are then used to derive the *significance probability* that each suspected drift satisfactorily explains difference between model predictions and measured responses [10]. This statistical notion of the significance probability is directly analogous to the Dempster-Shafer concept of evidential plausibility [11]. Therefore, the plausibilities derived from solving the regression models in reverse can be unambiguously combined with existing evidence using the methods described in Chapter 2.

5.3.4.1 Solving the Regression Equations in Reverse

In order to illustrate this technique, consider a manufacturing process with q observable responses controlled by p independent settings. Further, q regression models for the responses each contain up to p^* different terms. The regression equations are:

$$\mathbf{Y} = \mathbf{Z}'\hat{\Theta} + \mathbf{E} \tag{5.8}$$

where $\mathbf{Y}$ is the $n \times q$ array of the q responses collected from n experiments, $\mathbf{Z}$ is the $p^* \times n$ array of the values of the terms used in the equations, $\hat{\Theta}$ is the $p^* \times q$ array of the regression coefficients and $\mathbf{E}$ is the $n \times q$ array of residuals.

In general, the regression equations are nonlinear expressions of the process settings. However, they may be linearized using the truncated Taylor series expansion of $f(\mathbf{x})$:

$$f(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{z}'\hat{\Theta} + (\partial \mathbf{z}/\partial x_i)'\hat{\Theta}\Delta\mathbf{x}_i \tag{5.9}$$

where $\mathbf{x}$ is the column vector of the p process settings, $\mathbf{z}$ is the column vector of the p^* nonlinear expressions of the settings (i.e. - x_1x_2, x_3^2, etc.), $(\partial \mathbf{z}/\partial x_i)$ is the $p^* \times 1$ column vector of the partial derivatives of the regression terms of the ith setting, and Δx is a vector of small shifts in the each of the settings. Diagnosis occurs after the responses from m identical process runs have been observed. If these observations do not agree the model predictions, then a shift in one of the set-

tings is suspected.

A new value for the ith process setting may be extracted by solving the regression equations in "reverse" through minimizing the sum of squares of the residual:

$$\min(\Delta x_i \hat{\Theta}'(\partial \mathbf{z}/\partial x_i) - \bar{\mathbf{y}} + \hat{\Theta}'\mathbf{z})^2 \tag{5.10}$$

where $\bar{\mathbf{y}}$ is the $q \times 1$ column vector of the average process response over the m runs. The value of Δx_i which minimizes this expression is:

$$\Delta \hat{x_i} = \frac{(\partial \mathbf{z}/\partial x_i)'\hat{\Theta}\bar{\mathbf{y}} - \mathbf{z}'\hat{\Theta}\hat{\Theta}'(\partial \mathbf{z}/\partial x_i)}{(\partial \mathbf{z}/\partial x_i)'\hat{\Theta}\hat{\Theta}'(\partial \mathbf{z}/\partial x_i)} \tag{5.11}$$

As the subscript i implies, this shift is calculated for each of the p process settings.

5.3.4.2 Validating the Solution of the System - Test A

All malfunctions are not directly traceable to a shift in one of the process settings. In fact, even when a change in a setting is responsible for a malfunction, it could conceivably be of such magnitude that the linearized empirical models cannot adequately describe it. In such cases the solution of the regression system will offer no additional evidence since the residual defined in (5.10) and (5.11) will be too large to be distinguished from the error introduced by linearization. On the other hand, if the solution is indeed a valid one, then the residual will be normally distributed around zero. The $q \times 1$ vector of the residuals is defined for each of the p suspected process shifts as:

$$\xi_i \equiv \Delta \hat{x_i} \hat{\Theta}'(\partial \mathbf{z}/\partial x_i) - \bar{\mathbf{y}} + \hat{\Theta}'\mathbf{z} \tag{5.12}$$

The hypothesis test that determines whether these residuals are negligible or significant requires iterative use of the *student-t* statistic [10]. However, all of the q process responses are correlated, and an estimate of the $q \times q$ covariance matrix S is available from the original modeling experiment. This covariance matrix was estimated with $k \equiv n - p^*$ degrees of freedom. As a result,

the iterative use of the *student-t* test may be replaced by a single hypothesis test employing *Hotelling's* T^2 statistic with k degrees of freedom [12]. The value of the T^2 statistic, which is distributed according to the T^2 distribution, is:

$$T_A^2 = k\xi_i' S^{-1}\xi_i \tag{5.13}$$

The T^2 distribution is related to the well-known F distribution as follows:

$$T_{\alpha,q,k-q}^2 = \frac{q(k-1)}{k-q} F_{\alpha,q,k-q} \tag{5.14}$$

The residual ξ_i is *not* significantly different from zero and $\Delta\hat{x}_i$ is a "good" solution of the system at an α_A level of significance if:

$$T_A^2 < \frac{q(k-1)}{k-q} F_{\alpha_A,q,k-q} \tag{5.15}$$

Satisfaction of this inequality indicates that the malfunction *can* be explained in terms of a shift in the i th process setting. The maximum value of α_A that satisfies the inequality is known as the *significance probability* of the test. The significance probability is a measure of the "goodness" of the solution because it asymptotically approaches 0.0 when the residual is large and approaches 1.0 when the residual is small.

5.3.4.3 Determining the Significance of the Shift - Test B

Test A above validates the various solutions of the reverse regression equations by determining the relative capabilities of the calculated shifts $\Delta\hat{x}_i$ to account for the differences between the model predictions and experimental results. However, even after validating the solutions, it still remains to be seen if the actual shifts Δx_i are statistically significant. This is determined by means of another test based on the estimated shifts. The variable ζ_i is defined:

$$\zeta_i \equiv \Delta\hat{x}_i \hat{\Theta}'(\partial z/\partial x_i) \tag{5.16}$$

This variable is the non-centrality vector under the assumption that the actual shift is zero. It is derived from (5.12) by setting $\bar{y} = \hat{\Theta}'\bar{y}$. A second T^2 test can be performed based on:

$$T^2 = k\zeta_1' S^{-1}\zeta_1 \tag{5.17}$$

The maximum significance probability α_B of this statistic may be found in the same manner as (5.14) and (5.15). The value $(1-\alpha_B)$ measures the significance of the observed shift since α_B asymptotically approaches zero when the estimated shift is large and approaches one when the estimate is small.

5.3.4.4 Generating Evidential Plausibility from Tests A and B

The objective of this section is to use the statistical tests of the two previous sections to derive evidential plausibility of potential equipment faults. This requires acknowledgement of the fact that the statistical significance probability is equivalent to one minus the plausibility of a given event. In statistical terms, if an event is accepted at a 0.05 level of significance, then there exists at most a 5% chance of making a mistake accepting this event as true. Under these circumstances, using Dempster-Shafer terminology, the evidence that the event did not actually happen gives 0.05 "units" of support on a scale from 0.0 to 1.0. Therefore, the plausibility of the event is 0.95 (assuming zero uncertainty).

Using the regression models requires two tests to derive evidential plausibility. The first (Test A) establishes with a significance probability α_A that the residuals of the reverse regression equation are zero. Similarly, the second test (Test B) establishes with significance probability $1-\alpha_B$ that the calculated shift in a given process setting is statistically significant. For diagnosis, both of these assertions are combined and the plausibility p' that a given parameter has shifted is:

$$p' = \alpha_A(1-\alpha_B) \tag{5.18}$$

At this point, the role of evidential uncertainty u deserves some discussion. Since uncertainty is not a probabilistic concept, it is not possible to derive its value from the formal statistical tests above. In addition, some care must be exercised in any method for choosing a value for u since too large an uncertainty will result in negative evidential support if u is subtracted from p' directly (see § 2.3.5). In order to prevent this from occurring, the factor β is defined as:

$$\beta \equiv \frac{u}{1-u} \tag{5.19}$$

Then the final value of plausibility p which guarantees positive support is given by:

$$p = \frac{p' + \beta}{1 + \beta} \tag{5.20}$$

Since the chosen value of uncertainty is the same for all process settings modeled by the regression equations, the relative ranking of the suspected faults is unaffected by the introduction of β into the analysis.

It is also important to note that the derivation of evidential plausibility using the statistical tests outlined above yields a final relationship between plausibility and the inferred parametric shift that has a nearly sigmoid exponential form. This is a direct result of the exponential sigmoid shape of the *student-t* distribution [10-12].

5.3.4.5 Example of In-line Malfunction Diagnosis

In order to verify the above methodology, a simulated experiment was performed using the regression models which describe the Lam Autoetch plasma etcher. In this experiment, the in-line measurements of 50 wafers were generated via a random number generator. These measurements were generated around mean values given by the predicted responses of the optimized etch recipe from § 4.5.1 with standard deviations given by the one-sigma replication errors of the individual models. This data was used to calculate $\bar{y}$ in equations (5.10-12). A 5 sccm leak in the

CCl_4 flow was then simulated by using a value of only 145 sccm for this parameter in the z and $\partial z/\partial x_i$ vectors rather than the 150 sccm value specified by the recipe.

Table 5.1: Ranked Fault List for Simulated CCl_4 Leak

Rank	Parameter	Plausibility
1	CCl_4 flow	0.3265
2	RF Power	0.2459
3	O_2 flow	0.1953
4	*He* flow	0.1942
5	Gap	0.1744
6	Pressure	0.1227

Using the BLSS [13] statistical software package to calculate α_A and α_B yielded the fault ranking shown in Table 5.1. These plausibilities were calculated with an assumed uncertainty value of 10%. As is indicated by this list, the CCl_4 problem was correctly identified as the most likely candidate to explain the discrepancy between the model predictions and the simulated measurements. In general, this methodology has proven to work well, provided that the faults to be diagnosed are neither too small (in which case they produce statistically insignificant effects) or too large (preventing the linearized process models from adequately describing them).

This approach has been implemented to generate Dempster-Shafer evidential plausibility for possible faults related to each of the six process parameters in the etcher regression models. The support from in-line diagnosis is then combined with that from the two previous phases to produce the final diagnostic conclusions.

5.4 Distribution of Support Among Multiple Fault Hypotheses

Given any piece of diagnostic evidence, either the support generating function given by (5.1) or the method of solving regression equations in reverse described in § 5.3.3 above may be used to derive an initial probability mass distribution for a given knowledge source. BPMs are

assigned to the fault hypothesis sets H^+, H^- (see § 2.5), and H^o (the "no-fault" condition) in the following manner:

$$m<H^+, H^-, H^o, \theta> = <b, 0, 1-b-u, u>, \quad \varepsilon > 0 \tag{5.21}$$

$$m<H^+, H^-, H^o, \theta> = <0, b, 1-b-u, u>, \quad \varepsilon < 0 \tag{5.22}$$

The evidential intervals implied by this support assignment technique indicate that when the constraint residual is greater than zero, the probability that the constraint is high is between b and $b+u$, whereas the probability that it is low is between zero and u. When the residual is less than zero, these intervals are reversed. Note that the uncertainty involved in evaluating the constraint is assigned directly to θ. This means that u supports all relevant hypotheses and cannot be attributed to any particular subset of the frame of discernment. Assorted BPMDs for all knowledge sources are subsequently combined in the manner described in Chapter 2.

As an example, consider this support distribution scheme as applied to the frame of discernment discussed in § 2.4. This frame is repeated below for the sake of convenience:

A ≡ Mass Flow Controller Miscalibration
B ≡ Gas Line Leak
C ≡ Routing Valve Malfunction
D ≡ Incorrect Sensor Signal
E ≡ The "No-Fault" Condition

Let H^+ = (A $\cup$ B, C) for the violation of a particular constraint. This means that this set of violated faults (induced from $\varepsilon > 0$) contains faults A $\cup$ B or fault C. Such a scenario is possible when the particular evidence source cannot distinguish specifically between the fault subset "A or B" (mass flow controller miscalibration or gas line leak) and fault C (routing valve malfunction) alone. Furthermore, suppose that the records contained in the equipment maintenance database reveal that the set A $\cup$ B is responsible for the observed evidence 80% of the time when $\varepsilon > 0$, while C is responsible only 20% of the time. Thus, set A $\cup$ B gets $0.8*s(\varepsilon)$ and fault C gets $0.2*s(\varepsilon)$ (from 5.1) when $\varepsilon > 0$. Let H^- = D for this constraint.

In addition, consider a second evidential constraint whose violation yields $H^+ = B$ and $H^- = (A \cup C)$. Further, assume that $s_1(\varepsilon_1) = 0.6$ when $\varepsilon_1 > 0$ for the first constraint and $s_2(\varepsilon_2) = 0.7$ when $\varepsilon_2 < 0$ for the second constraint. Then, the following support distribution results:

$$m_1 \langle A \cup B, C, D, (A \cup B \cup C \cup D)', \theta \rangle = m_1 \langle A \cup B, C, D, E, \theta \rangle = \langle 0.48, 0.12, 0, 0.2, 0.2 \rangle \quad (5.23)$$

$$m_2 \langle B, A \cup C, (A \cup B \cup C)', \theta \rangle = m_2 \langle B, A \cup C, D \cup E, \theta \rangle = \langle 0, 0.7, 0.1, 0.2 \rangle \quad (5.24)$$

This illustrates the manner in which the BPMDs combined in § 2.4 are initially derived. Similar methods are applied to all evidence sources in the system to obtain a complete mapping of support from the evidence space to the fault space (or frame of discernment).

5.5 Summary

This chapter has outlined a paradigm for the generation of Dempster-Shafer evidential support during three chronological phases of malfunction diagnosis. These three phases are based on the availability of evidence during the maintenance, on-line, and in-line portions of equipment operation. Each phase of diagnosis is accompanied by a unique method of support generation based on statistical principles as well as heuristics.

In conjunction with the operational equipment description contained in Chapter 3 and the quantitative equipment models developed in Chapter 4, the support generation techniques in this chapter complete the set of components necessary to implement the automated malfunction diagnosis system on the Lam plasma etcher. The details of that implementation are the topic of the next chapter.

References for Chapter 5

[1] G. Shafer, "A Mathematical Theory of Evidence," *Princeton University Press*, 1976.

[2] M. A. Kramer, "Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems," *Journal of the American Institute of Chemical Engineers*, vol. 33, no. 1, January, 1987.

[3] G. S. May and C. J. Spanos, "Automated Malfunction Diagnosis of a Plasma Etcher," *Proceedings of the 1991 International Semiconductor Manufacturing Science Symposium*, May, 1991.

[4] N. H. Chang and C. J. Spanos, "Chronological Equipment Diagnosis with Evidence Integration: An LPCVD Application," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 1, February, 1991.

[5] N. H. Chang, "Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB/ERL M90/61*, May, 1990.

[6] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, April, 1987.

[7] D. Mudie and N. H. Chang, "FAULTS: An Equipment Maintenance and Repair System Using a Relational Database," *Proceedings of the 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium*, October, 1990.

[8] H. Guo, C. J. Spanos, and A. Miller, "Real Time Statistical Process Control for Plasma Etching," *Proceedings of the 1991 International Semiconductor Manufacturing Science Symposium*, May, 1991.

[9] D. C. Montgomery, *Introduction to Statistical Quality Control*, 2nd Edition, New York: Wiley, 1989.

[10] G. E. P. Box, W. B. Hunter, and J. S. Hunter, *Statistics for Experimenters*, New York: Wiley, 1978.

[11] C. J. Spanos and G. S. May, "Using Regression Equations for Model-Based Diagnosis in the Berkeley Computer-Aided Manufacturing System," *Workshop on Intelligent Diagnostic and Control Systems for Manufacturing*, Boston, July, 1990.

[12] R. J. Harris, *A Primer of Multivariate Statistics*, San Diego: Academic Press, 1975.

[13] D. M. Abrahams and F. Rizzardi, *BLSS: The Berkeley Interactive Statistical System*, New York: Norton, 1988.

CHAPTER 6

SOFTWARE IMPLEMENTATION OF DIAGNOSTIC METHODOLOGY

6.1 Introduction

In recent years, software design methods have undergone a change in philosophy toward what is known as "object-oriented programming" (OOP) [1]. In an effort to obtain improvement in software productivity and reusability, this style of programming has been selected as the preferred technique for the Berkeley Computer-Aided Manufacturing (BCAM) system (see Chapter 1). Consequently, this style of programming has also been adopted for the realization of the malfunction diagnosis system. The C++ programming language [2] has been chosen to implement the object-oriented approach. In addition to some background information on OOP, this chapter provides a description of the C++ software which implements the diagnostic methodology and an overview of the flow of information within the system.

6.2 Object-Oriented Programming

The central philosophy behind OOP is the concept that the design of an application should be modeled as closely as possible to the real-world objects which are embodied in the application itself. This approach differs from structured programs which are organized as a small number of relatively large and complex procedures that manipulate a small number of large and complex data structures. Although structured programs are suitable for development early in the design cycle, they can be difficult to maintain throughout the lifetime of the software. This is especially true when programmers are required to make extensive changes during implementation that were not forseen during the initial prototyping phase [1].

Object-oriented programs extend the structured approach by combining the principle of *data abstraction* with traditional concepts of hierarchical structure and modularity. Each

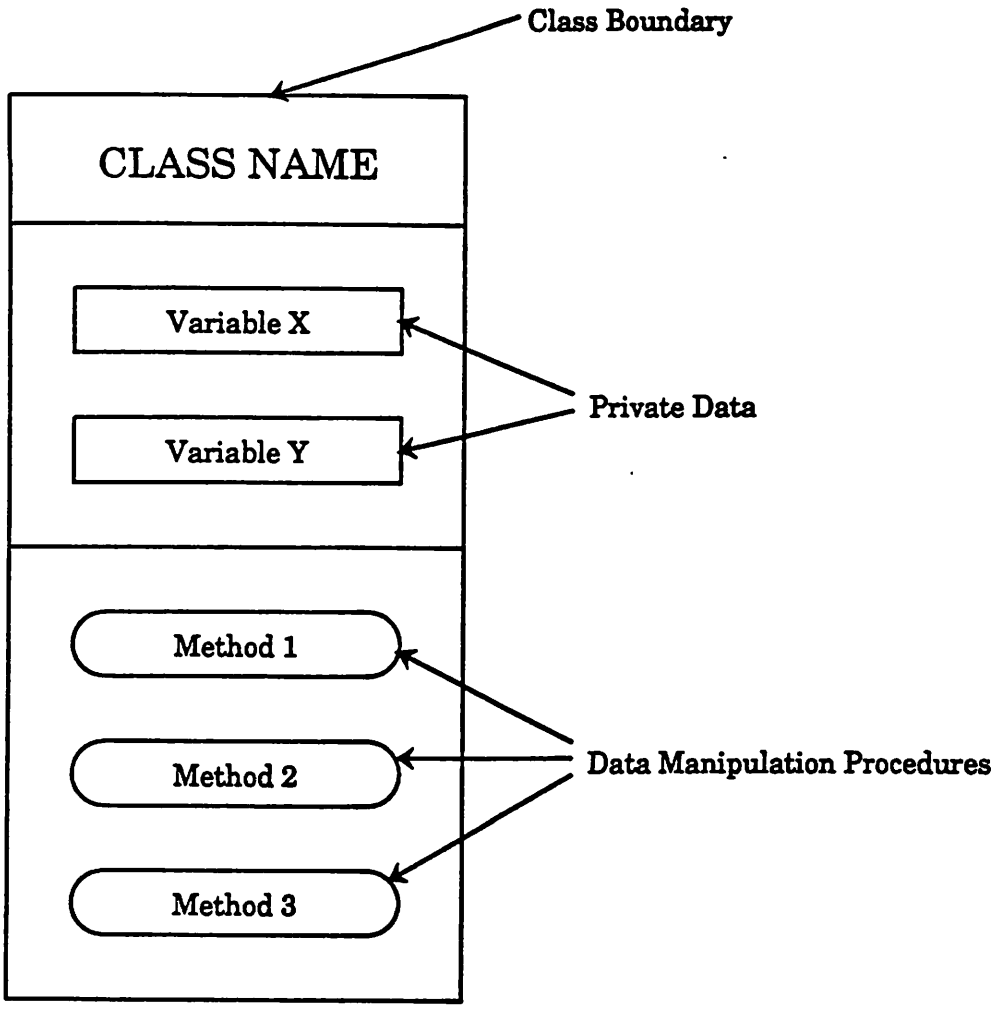


Figure 6.1 - Illustration of generic OOP objects.

individual unit of data abstraction, or *object*, has a limited number of permissible operations which control the data within its boundaries. The data within the boundary of one object is completely inviolate from the point of view of other objects, thereby preventing interdependencies between objects. The procedures which operate on data within an object, known as *methods*, are the only means of accessing this data. Some methods are *public* (accessible from outside the object), while others are *private* (accessible only to other object methods). Objects interact by sending and receiving messages without specific knowledge of how the calling procedure is

implemented. The object structure is illustrated in Figure 6.1.

Aside from data abstraction and message passing, other significant features unique to OOP include: *polymorphism*, *inheritance*, and *class hierarchy*. Polymorphism is the highly desirable ability to use the same function name to refer to different function implementations. For example, polymorphism would allow the same "read" function to input character or numeric data. Inheritance is the mechanism by which objects of one class may possess the properties of another. Using this property, polymorphic methods in a subclass override their counterparts in the parent class, and changes to methods in the parent are automatically inherited by all descendent classes. Inheritance allows the programmer to organize class objects into a hierarchy in which generic high-level classes are refined into successively more specific subclasses. In this way, classes may be arranged just like a zoological taxonomy in which animals are categorized by their similarities and differences. For example, a "mammal" class may have subclasses "dog" and "horse".

6.3 Class Structure and Associated Methods for Diagnostic Objects

The following section describes the class structure of the equipment diagnosis system. Three major classes are required to implement this methodology: 1) the *BPM* class; 2) the *evidence* class; and 3) the *fault-set* class. These three classes form the basis of all evidence combination and belief distribution in the system. Each is presented along with an explanation of their associated methods.

6.3.1 The BPM Class

Initially, an object-oriented dialect of Common Lisp known as CLOS was selected as the language with which to implement the Dempster-Shafer rules of evidence combination in the diagnostic system [3-5]. However, the computational speed and efficiency of interpreted languages such as Lisp or Prolog [6] leaves much to be desired. This is a critical deficiency since

the Dempster-Shafer techniques require a significant number of computations, and real-time plasma etch diagnosis requires that these calculations take place at a pace commensurate with that of the etching itself, which is on the order of minutes [7].

Due to the inadequacies of CLOS, the C++ programming language (an object-oriented superset of C) has been implemented to increase computational speed. The expected increase was justifiable since C is a compiled, rather than interpreted language. As a result, C++ employs much more efficient memory management techniques than any dialect of Lisp [2]. In fact, after the code was written and tested, C++ was found to have more than a factor of 1000 advantage in CPU time over CLOS for functions performing analogous tasks (1.06 sec for CLOS versus less than 1.0 msec for one BPMD combination in C++).

In this implementation, BPMDs are stored as linked lists of C++ objects called "bpm_nodes". The exact format of these objects is described in Appendix 6.1a. Each bpm_node contains the following data:

1) Pointers to the head of the list (*head*), the address of the current node (*cur*), and the next node in the linked list (*next*).

2) A stream of bits which represent the presence or absence of certain faults in the frame of discernment (*bits*).

3) The support (*belief*) and plausibility (*plaus*) attributed to the particular fault set described by this node.

4) A flag indicating whether or not this node contains the belief ascribed to θ (*theta*).

The structure of the linked list is visualized in Figure 6.2, which depicts the BPMD m_1, which was discussed in § 2.4.

The bit stream merits further explanation. Essentially, each set of faults is assigned to a particular stream, and each fault within a given set is mapped to a particular bit within the stream. Consider, the frame of discernment in § 2.4. This frame would be represented by a stream of five

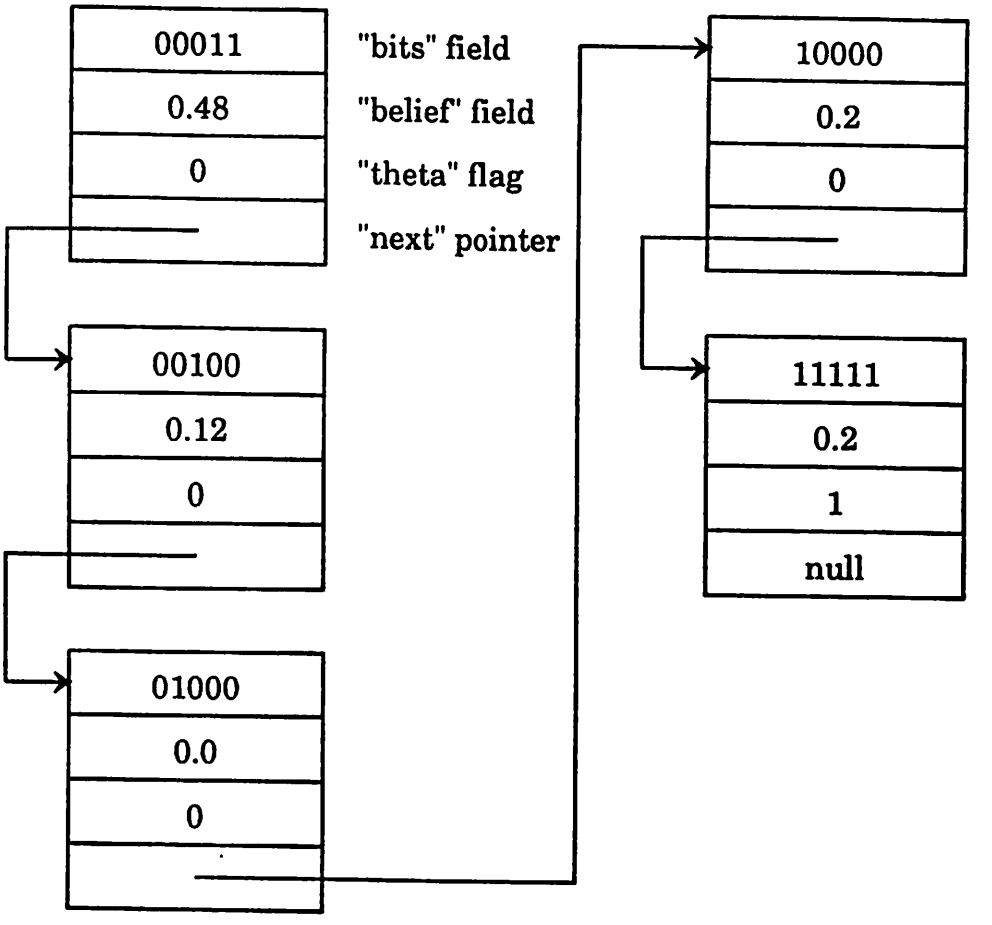


Figure 6.2 - The linked list structure used to implement BPMDs in C++.

bits, one for each of the five faults. The least significant bit is assigned to fault A. For example, the *bits* field in the bpm_node assigned to fault set A $\cup$ C would be *00101*. Likewise, fault set A $\cup$ B would be represented as *00011*, fault D would be *01000*, and so forth. When BPMDs are combined, the *bits* field of individual bpm_nodes are bitwise logically "AND"-ed in order to determine the belief to be attributed to their intersection as in equations (2.8) and (2.9).

In addition to the data indicated above, the bpm_node object (or "class") also contains various methods which are used to manipulate the linked lists in order to implement the Dempster-Shafer methodology. The most critical functions in terms of performing Dempster-Shafer evi-

dence combination are the public (or "friend") functions "arb_bpmd_comb" and "bel_pls_encript". The "arb_bpmd_comb" function combines BPMDs from two arbitrary evidence sources, and "bel_pls_encript" calculates the evidential intervals of support and plausibility resulting from this combination. Most of the other methods in this class are designed to perform various subtasks related to combining the BPMDs and obtaining the evidential intervals. The remaining methods are: 1) a function which appends new bpm_nodes to the linked list; 2) standard C++ functions to allocate and deallocate memory space for the class (called *constructors* and *destructors*); and 3) methods used to step through nodes in the linked list [2]. For a more detailed description of each method, refer to Appendices 6.1a and 6.2.

6.3.2 The Evidence Class

The evidence class contains information related to all forms of evidence for maintenance, on-line, and in-line diagnosis. Like the BPM class, this class also utilizes a linked list data structure. This structure appears in Appendix 6.1b. In addition to the usual list pointers, each node of the evidence linked list (called an "evidence_node") contains the data below:

1) The name of this particular piece of evidence (*name*).

2) The uncertainty associated with this piece of evidence (*uncertainty*).

3) The "sharpness" parameter of the support function (*bel_g*).

4) The name of the piece of equipment corresponding to this evidence (*equip*).

For on-line evidence, two additional slots are defined:

5) The recipe setpoint for this piece of evidence (*setpt*).

6) The tolerance limit of this evidence (*tol*).

Overall, the program contains three distinct evidence lists, one for each of the three phases of diagnosis. As an example, a portion of the on-line evidence list is depicted in Figure 6.3. This

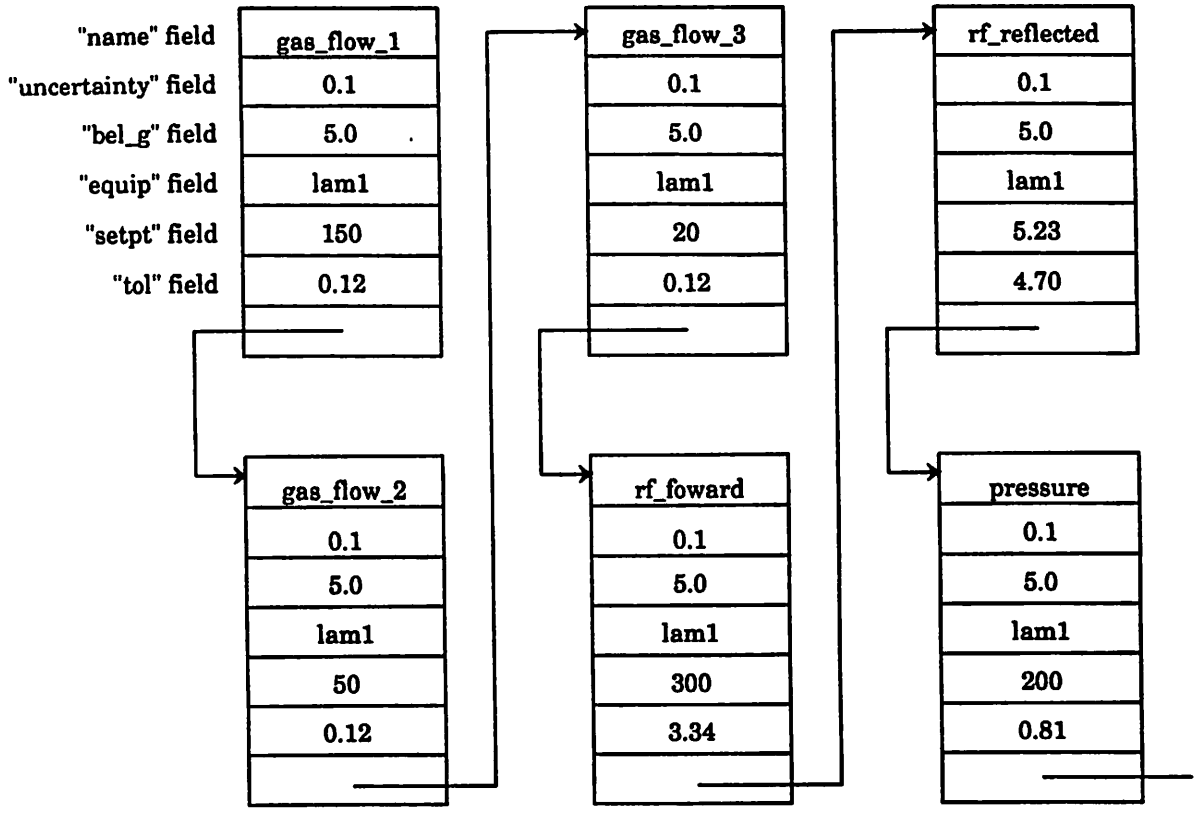


Figure 6.3 - The on-line evidence list structure.

figure shows a portion of the on-line evidence list. Nodes for each of the real-time data channels (see § 4.2) are chained together to form this list [8-9]. A similar approach is taken to construct the maintenance and in-line evidence lists.

The three evidence lists are initialized by the public functions "initialize_maintenance_evidence", "initialize_online_evidence", and "initialize_inline_evidence." Another public method, "inline_belgen", generates Dempster-Shafer evidential plausibility for in-line evidence in the manner described in § 5.3.4. Aside from standard methods used to append and iterate through the linked list of evidence nodes described

in the previous section, this class has only one private function. This function (which is called "get_online_spc_info") retrieves the values for the *setpt* and *tol* slots for the on-line evidence list. For a more thorough explanation of these functions, see Appendices 6.1b and 6.2.

6.3.3 The Fault-Set Class

The diagnostic software uses yet another type of linked list, the *fault-set*, to quantify the knowledge base of the plasma etcher (see Chapter 3). The contents of the fault-set lists are outlined in the definition of the "fault_set_node" class in Appendix 6.1c. Aside from list pointers, each node contains the following data:

1) The name of the piece of evidence associated with this fault-set (*evi_name*).

2) A flag indicating whether this fault-set corresponds to positive or negative constraint violation (*level*).

3) The number of faults in the set (*fault_cnt*).

4) A symbolic list of equipment faults belonging to the set (*symbols*).

5) The bit stream representation (as described in § 6.2.1) of the fault symbols (*bits*).

6) The proportion of belief attributed to this set (*frac*).

The structure of the fault-set linked list appears in Figure 6.4. The fault-set depicted in this figure corresponds to the one described in § 5.4. Here, the *level* flag is set to 1 for positive constraint deviations and -1 for negative deviations. Each fault symbol corresponds to a single component in the frame of discernment. The *frac* variable is the fraction of the generated support attributed to the fault-set in a single node of a fault-set list. This variable represents the weight that the set is given in the diagnosis of a particular violated constraint [10].

In addition to the above data and the standard C++ functions to implement linked lists (see § 6.2.1 above), the fault_set_node class also contains several methods used to manipulate fault-sets for diagnosis. The three most important functions associated with the fault_set_node class

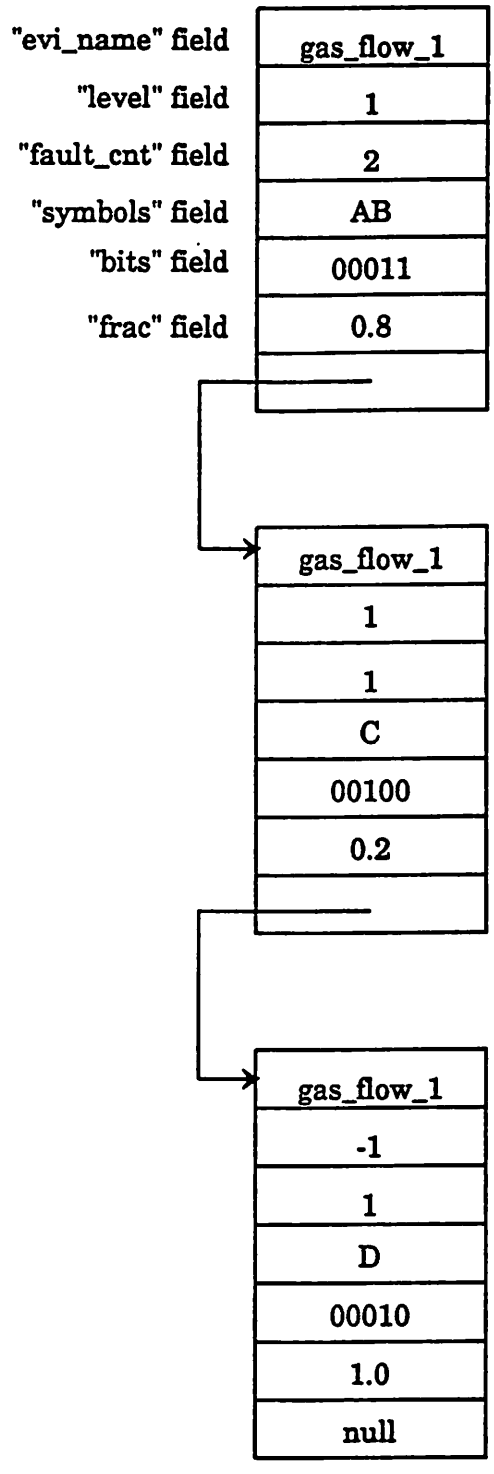


Figure 6.4 - The linked list structure which implements fault-sets.

are the public functions "initialize_fault_sets", "make_fault_set" and "make_bpmd". The "initialize_fault_sets" function creates a global fault-set which contains every fault combination in the frame of discernment. From this overall fault list, "make_fault_set" constructs a subset of this list for a particular piece of evidence. This evidence originates from the maintenance, on-line and in-line evidence lists (refer to § 6.2.2). Finally, "make_bpmd" creates the BPMD which corresponds to a fault-set for a given direction of constraint violation, numerical support and uncertainty value. For further information regarding the methods attached to the fault_set_node_class, see Appendices 6.1c and 6.2.

6.4 Diagnostic Algorithms and Information Flow

The BPM, evidence, and fault-set linked lists are the basic data structures that are manipulated during the execution of the diagnostic software. This section provides an overview of the manner in which program execution takes place. Figure 6.5 shows the global flow of information within the diagnostic system. In this figure, rectangular boxes symbolize equipment, cylinders indicate data or databases, and circles represent software. The direction of arrowheads is indicative of the direction of information flow. The primary body of code is divided into three modules which reflect the three stages of diagnosis. MAINT performs maintenance diagnosis. ONLINE implements both maintenance and on-line diagnosis. INLINE initiates all three diagnostic phases. The algorithms used to implement these three modules are described below. Execution of the entire diagnostic package requires: the etch recipe, the real-time etch data and the in-line measurements. The system produces two forms of output: a text file containing a ranked list of equipment malfunctions or plots of the Dempster-Shafer belief accumulated by each component fault versus time.

Although it has only been applied to a plasma etcher in this dissertation, most of the structure of the diagnostic system is generic and may accomodate other semiconductor manufacturing

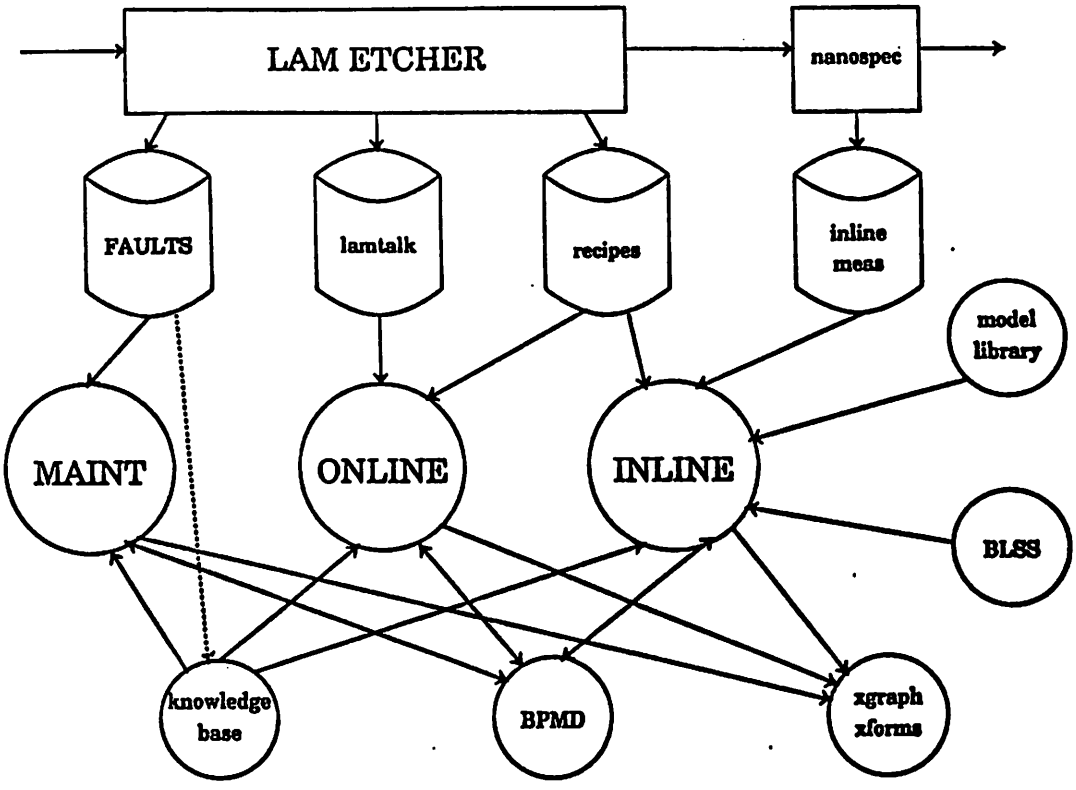


Figure 6.5 - Information flow within the diagnostic system.

processes. The only portions of the code which must be significantly modified for new equipment applications are the knowledge base and the equipment model library. As part of the overall BCAM architecture (see Chapter 1), the system also works closely with several other CAM functions, including the statistical process control module, which generates alarms and initiates diagnosis. In addition, as has been previously mentioned, the system interacts with both the equipment monitoring and modeling functions.

6.4.1 Maintenance Diagnosis

The maintenance diagnosis module (MAINT) begins execution by creating an overall BPMD and storing it in a "record-oriented" input/output file [2]. This BPMD is used for fast tem-

porary storage of BPMDs as they are combined throughout the maintenance diagnostic process. The file is initialized by setting each fault's support to zero and plausibility to one, signifying that no diagnostic information is available at this point. Next, both the maintenance evidence list and the global fault-set are initialized as described in § 6.2.2 and 6.2.3 above.

Following these initialization procedures, MAINT loops through each piece of evidence in the maintenance evidence list, continuously updating and storing the overall BPMD each case. Belief for faults in the maintenance phase of diagnosis is obtained by comparing the mean-time-to-failure of each equipment component to the elapsed time since its last failure (see § 5.3.2). This component failure information is read from a file which is generated by FAULTS, the computerized equipment maintenance record-keeping system [11].

After all components in the maintenance evidence list have been processed, a text file is created which contains the ranked list of all equipment faults resulting from maintenance diagnosis. This file contains the name of each component along with its final support and plausibility values (see Figure 6.9 below). However, faults are ranked according to their support value only. Updating the fault ranking file concludes the maintenance phase.

6.4.2 On-line Diagnosis

Unlike the maintenance module, the on-line diagnosis module (ONLINE) requires two forms of user input to initiate execution: the name of a file which contains the etch recipe and a second file that contains the real-time sensor readings of etcher operating conditions from the equipment monitoring system. This data is transmitted over the local area network via the SECS-II protocol by a program called LAMTALK [9]. The recipe file is obtained from the BCAM recipe database [12]. The format of this file is illustrated by Figure 6.6.

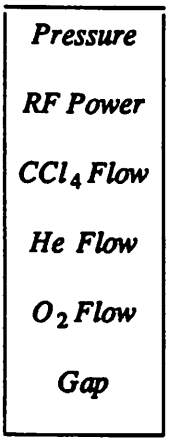


Figure 6.6 - Format of recipe file.

The initialization portion of ONLINE is similar to the maintenance module, except that this module defines an additional record-oriented BPMD file to serve as temporary storage for on-line BPMDs as well as an on-line evidence list. After initialization, maintenance diagnosis takes place as described in the previous section. Next, the sensor data file is read and the standard deviations of the sensor readings are obtained by the "get_online_spc_info" function described in § 6.2.2 above.

ONLINE performs diagnosis by continuously updating the on-line BPMD through the examination of all sources of evidence in the on-line evidence list. For each piece of evidence in this list, fault belief is generated for each of the eleven sensor readings (refer to § 4.2). These readings are available in sets that are taken at a particular instant in time. Each set is called a "sample". Usually, LAMTALK monitors and stores one sample per second.

Evidential support is generated using the CUSUM [13] scheme described in § 5.3.3. After the cumulative sums are calculated and belief is generated, BPMDs are combined for each set of concurrent readings. The overall BPMD from each set is stored in individual files which correspond to component faults. The format of these files is compatible with XGRAPH [14], which is used to produce support versus time plots for each individual fault (refer to Figure 6.9

below). The first data point in each of these plots corresponds to the support calculated during maintenance diagnosis. All support versus time data is stored in a separate "belief" directory. In addition to these graphs, ONLINE also updates the maintenance fault ranking.

6.4.3 In-line Diagnosis

The module which implements in-line diagnosis (INLINE) uses the same input as ONLINE. However, aside from the etch recipe and sensor data, INLINE also requires the name of the file that contains a summary of the in-line measurements for a particular lot of wafers. The format of this file appears in Figure 6.7. During initialization, INLINE defines another record-oriented BPMD file as well as an in-line evidence list in addition to those files and evidence lists created in ONLINE. Afterwards, maintenance and on-line diagnosis are performed as described above.

In-line diagnosis is subsequently executed by looping through all sources of in-line evidence and continuously updating the in-line BPMD. INLINE uses the etch recipe and the BCAM equipment model library (see Appendix 4.2) to compare model predictions to the measurements in the in-line measurement file. This module interfaces with the BLSS [15] statistical package to generate evidential plausibility using the "reverse" regression approach of [16], which is explained in § 5.3.4.

Mean Etch rate
Mean Nonuniformity
Mean Oxide Selectivity
Mean Resist Selectivity

Figure 6.7 - Format of in-line measurement file.

The BPMDs which result from analyzing in-line evidence are appended to the XGRAPH files described in the previous section. Like ONLINE, INLINE also updates the maintenance ranking. Thus, upon the conclusion of in-line diagnosis, complete support versus time plots are available for all three diagnostic phases (see Figure 6.9). In each plot, the first data point corresponds to support derived during maintenance diagnosis and the last four points are obtained from performing in-line diagnosis for each of the four in-line measurements. All intermediate points result from the on-line phase.

6.5 Diagnosis User Interface

The executable files for maintenance, on-line and in-line diagnosis described above allow the execution of each phase independently. Specific instructions on how to do so directly are provided in Appendix 6.3. However, diagnosis may also be initiated through the more user-friendly BCAM interface [12]. This interface uses the *X Toolkit* [17] to call BCAM the various applications from an *X Window* environment [14].

Figure 6.8 shows the windows associated with each step in running the diagnostic software. Window (1) is the BCAM main menu. After the user enters his/her account name, the desired BCAM module may be selected with a mouse click. When diagnosis is selected, the dialogue boxes in window (2) prompt the user to select a particular piece of equipment to diagnose. The choice of "LAM1" causes the appearance of window (3), which allows the user to select the appropriate recipe file from the recipe editor shown in window (4). Next, window (5) permits the selection of either maintenance, on-line, or in-line diagnosis. Execution of on-line diagnosis requires the selection of the file containing the appropriate sensor data from window (6). Finally, initiating in-line diagnosis necessitates the choice of the appropriate in-line measurement file in window (7).

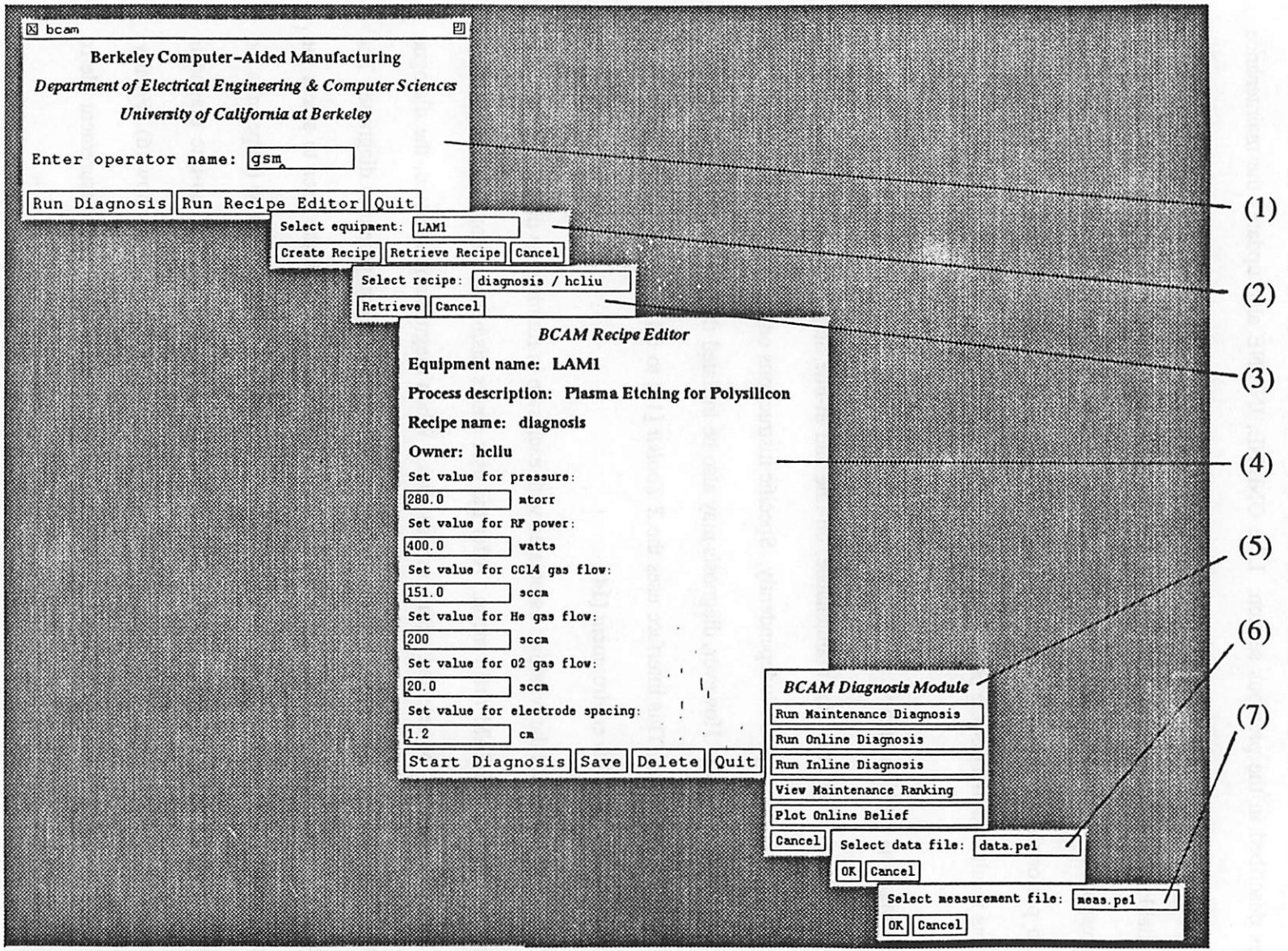


Figure 6.8 - Executing diagnosis from the BCAM user interface.

The output interface of the diagnosis module is illustrated in Figure 6.9. Window (1) of this figure shows the fault ranking file which results from maintenance diagnosis. If on-line or in-line diagnosis has been executed, the user may select a particular component from the menu in window (2) to produce support versus time graphs. Examples of such plots appear in windows (3) and (4).

6.6 Summary

The chapter has provided an overview of the software which implements the methodology for equipment malfunction diagnosis described in previous chapters. The diagnostic system has been programmed in an object-oriented fashion using the C++ language. It resides on a Sun-class UNIX workstation which may be linked directly to manufacturing equipment. The diagnostic system is one module of the overall BCAM architecture. As such, this system works closely with several other applications, including statistical process control, real-time monitoring, and equipment modeling.

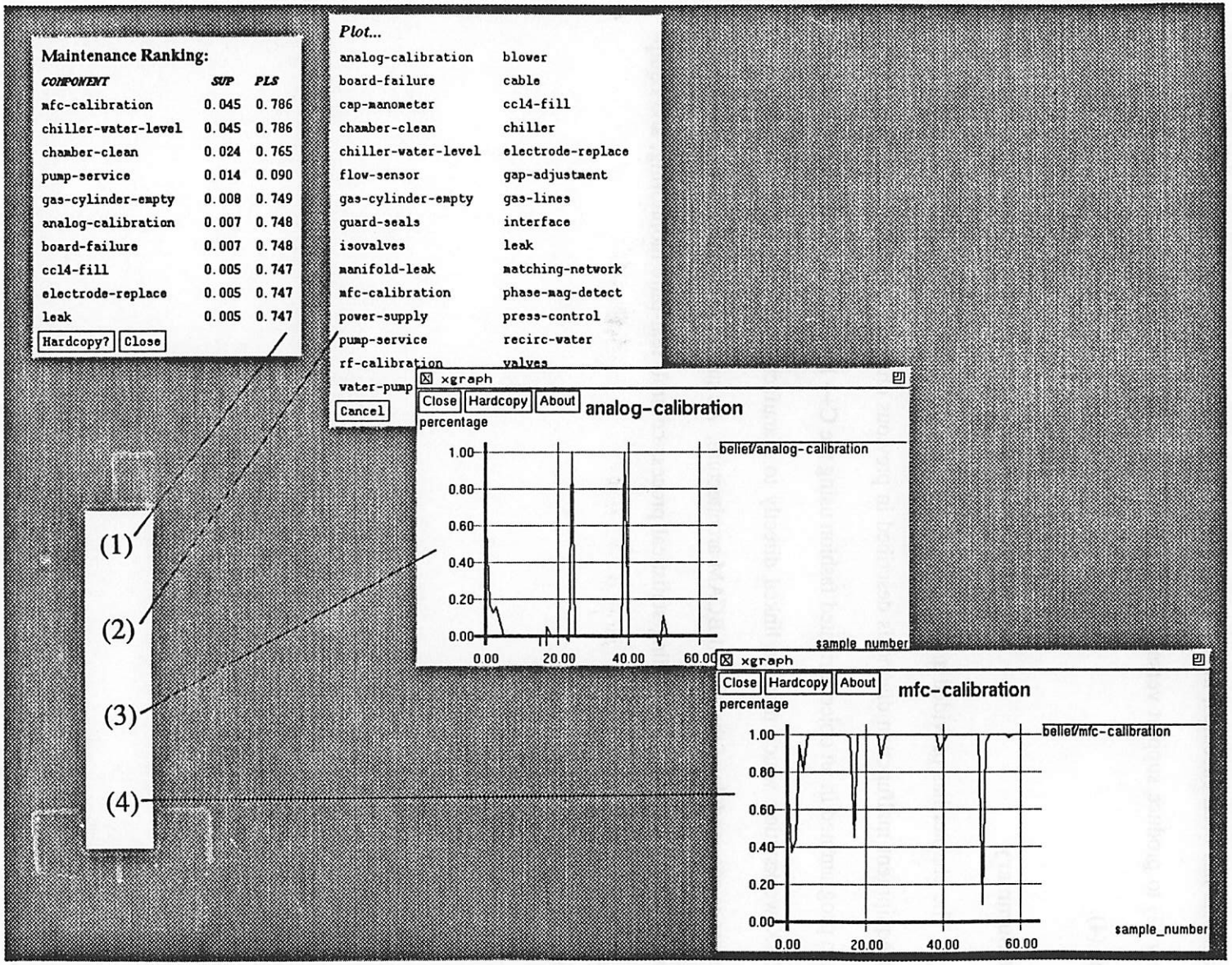


Figure 6.9 - Diagnosis output interface.

References for Chapter 6

[1] J. Bessemer, "An Introduction to Object-Oriented Design and Programming," *CASE Outlook*, 1987.

[2] K. Weiskamp and B. Flaming, *The Complete C++ Primer*, San Diego: Academic Press, 1990.

[3] S. A. Keene, *Object-Oriented Programming in Common Lisp*, Reading: Addison-Wesley, 1989.

[4] G. S. May and C. J. Spanos, "Automated Malfunction Diagnosis of a Plasma Etcher," *Proceedings of the 1991 International Manufacturing Science Symposium*, May, 1991.

[5] N. H. Chang and C. J. Spanos, "Chronological Equipment Diagnosis with Evidence Integration: An LPCVD Application," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 1, February, 1991.

[6] N. C. Rowe, *Artificial Intelligence through Prolog*, New Jersey: Prentice Hall, 1987.

[7] D. M. Manos and D. L. Flamm, *Plasma Etching: An Introduction*, San Diego: Academic Press, 1989.

[8] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, March, 1985.

[9] N. H. Chang, "Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB/ERL M90/61*, May, 1990.

[10] D. Mudie and N. H. Chang, "FAULTS: An Equipment Maintenance and Repair System Using a Relational Database," *Proceedings of the 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium*, October, 1990.

[11] H. Liu, *private communication*, March, 1991.

[12] D. C. Montgomery, *Introduction to Statistical Quality Control*, 2nd Edition New York: Wiley, 1989.

[13] V. Quercia and T. O'Reilly, *X Window System User's Guide*, Cambridge: O'Reilly & Associates, Inc., 1989.

[14] D. Abrahams and F. Rizzardi, *BLSS: The Berkeley Interactive Statistical System*, New York: Norton, 1988.

[15] C. J. Spanos and G. S. May, "Using Regression Equations for Model-Based Diagnosis in the Berkeley Computer-Aided Manufacturing System," *Workshop on Intelligent Diagnostic and Control Systems for Manufacturing*, July, 1990.

[16] A. Nye and T. O'Reilly, *X Toolkit Intrinsics Programming Manual*, Cambridge: O'Reilly & Associates, Inc., 1990.

APPENDIX 6.1a

CLASS DECLARATION AND ASSOCIATED METHODS

FOR BPMD OBJECTS

```
/*

  Diagnosis module of BCAM

  Copyright (c) 1990 Regents of the University of California

      Permission to use, copy, modify, and distribute this software and its documentation for any
      purpose and without fee is hereby granted, provided that the above copyright notice appear
      in all copies and that both that copyright notice and this permission notice appear in sup-
      porting documentation, and that the name of the University of California not be used in
      advertising or publicity pertaining to distribution of the software without specific, written
      prior permission. The University of California makes no representations about the suitabil-
      ity of this software for any purpose. It is provided "as is" without express or implied war-
      ranty.

  Author: gsm
  Source: /home/radon 1/bcam/src/diagnosis/lam
  Revision: 1.0
  Date: 90/05/24 11:58:37

  FileName: bpmd.h

*/


//*****************************************************************************


//
// Class declarations for bpmd nodes (nodes of linked lists).
//

// Bpm_node class. Each node contains the following data:
//      1) "bits" - its fault bit sequence.
//      2) "belief" - evidential support value.
//      3) "plaus" - evidential plausibility value.
//      4) "theta" - a flag to indicate whether or not the node is theta.
//
// In addition, each node contains pointers to the head of the list, the
// current node and next node in the list.
```

```
class bpm_node {
public:
        bpm_node *head;
        bpm_node *next;
        bpm_node *cur;
        long bits;
        float belief;
        float plaus;
        short theta;
        bpm_node(void) {        // constructor
                head = 0;
                theta = 1;
        }
        ~bpm_node(void); // destructor
        void reset_iter(void) {
                cur = head;             // reset the iterator
        }
        bpm_node *iter(void);           // iterator function
        void append(long b, float bel, short thet, float pls=0);
        void modify_bit_value(void);
        int my_length(void);
        float my_nth(int pos);
        void my_subst(int pos, float val);
        void normalize(float factor);
        friend bpm_node arb_bpmd_comb(bpm_node &b1, bpm_node &b2);
        friend bpm_node bel_pls_encript(int lngth, bpm_node &b);
        friend int bit_vec_equal(long stream, bpm_node *b);
        friend bpm_node *my_bit_and(bpm_node *n1, bpm_node *n2);
};
```

APPENDIX 6.1b

CLASS DECLARATION AND ASSOCIATED METHODS FOR EVIDENCE OBJECTS

```
/*

  Diagnosis module of BCAM

  Copyright (c) 1990 Regents of the University of California

      Permission to use, copy, modify, and distribute this software and its documentation for any
      purpose and without fee is hereby granted, provided that the above copyright notice appear
      in all copies and that both that copyright notice and this permission notice appear in sup-
      porting documentation, and that the name of the University of California not be used in
      advertising or publicity pertaining to distribution of the software without specific, written
      prior permission. The University of California makes no representations about the suitabil-
      ity of this software for any purpose. It is provided "as is" without express or implied war-
      ranty.

  Author: gsm
  Source: /home/radon 1/bcam/src/diagnosis/lam
  Revision: 1.0
  Date: 90/06/12 09:54:43

  FileName: evidence.h

*/


//*****************************************************************************

//
// Class declarations for evidence and its subclasses.
//

// Evidence_node class (foundation of all evidence). Each node contains the
// following data:
//      1) "name" - name of this piece of evidence.
//      2) "uncertainty" - the uncertainty in this evidence.
//      3) "bel_g" - the belief function parameter.
//      4) "equip" - the equipment name.
//      5) "setpt" - the recipe setpoint of this evidence (if applicable).
//      6) "tol" - 3-sigma tolerance of this evidence (if applicable).
//
// In addition, each node contains pointers to the head of the list, the
```

```
// current node, and the next node in the list.

class evidence_node {
public:
        evidence_node *head;
        evidence_node *next;
        evidence_node *cur;
        char *name;
        float uncertainty;
        float bel_g;
        char *equip;
        float setpt;
        float tol;
        evidence_node (void) {          // constructor
                head = 0;
        }
        ~evidence_node(void);           // destructor
        void reset_iter(void) {
                cur = head;             // reset the iterator
        }
        evidence_node *iter(void);      // iterator
        void append(char *na, float u, float g, char *eq);
        void get_online_spc_info(char *recipefile);
        friend evidence_node initialize_maintenace_evidence(void);
        friend evidence_node initialize_online_evidence(void);
        friend evidence_node initialize_inline_evidence(void);
        friend float inline_belgen(char *ename,float u,char *rfile,char *mfile);
};
```

APPENDIX 6.1c

CLASS DECLARATION AND ASSOCIATED METHODS FOR FAULT-SET OBJECTS

```
/*

  Diagnosis module of BCAM

  Copyright (c) 1990 Regents of the University of California

      Permission to use, copy, modify, and distribute this software and its documentation for any
      purpose and without fee is hereby granted, provided that the above copyright notice appear
      in all copies and that both that copyright notice and this permission notice appear in sup-
      porting documentation, and that the name of the University of California not be used in
      advertising or publicity pertaining to distribution of the software without specific, written
      prior permission. The University of California makes no representations about the suitabil-
      ity of this software for any purpose. It is provided "as is" without express or implied war-
      ranty.

  Author: gsm
  Source: /home/radon 1/bcam/src/diagnosis/lam
  Revision: 1.0
  Date: 90/06/12 09:54:43

  FileName: fault_set.h

*/

//
// Class declarations for symbolic fault set nodes (nodes of linked lists).
// Each fault set is associated with one piece of evidence.
//


//*****************************************************************************


// Fault_set_node class. Each node contains the following data:
//      1) "evi_name" - evidence name associated with this fault set.
//      2) "level" - a short integer flag which indicates whether constraint
//              violation is positive or negative (+/- 1).
//      3) "fault_cnt" - the number of faults in the fault set.
//      4) "symbols" - symbolic list of faults (such as 'ABD' or 'ADE').
//      5) "bits" - the bits corresponding to the above symbols
//      6) "frac" - the weighted belief fraction of the set.
```

```
//
// In addition, each node contains pointers to the head of list, the current
// node, and the next node in the list.

class fault_set_node {
public:
        fault_set_node *head;
        fault_set_node *next;
        fault_set_node *cur;
        char *evi_name;
        short level;
        int fault_cnt;                  // how many faults in this node?
        char *symbols;                  // fault symbols
        long bits;
        float frac;
        fault_set_node(void) {          // constructor
                head = 0;
        }
        ~fault_set_node(void);          // destructor
        void reset_iter(void) {
                cur = head;             // reset the iterator
        }
        fault_set_node *iter(void);     // iterator
        void append(char *evi, short lev, int fc, char *sym, float fr);
        void clear_fault_bits(void);
        int count_faults(void);
        void get_fault_bits(void);
        friend int char_to_int(char c);
        friend fault_set_node initialize_fault_sets(void);
        friend bpm_node make_bpmd(fault_set_node &fs, short lev, float bel,
                        float u, int flts);
        friend fault_set_node make_fault_set(char *ev, fault_set_node &ofs);
};
```

APPENDIX 6.2

DIAGNOSTIC SOFTWARE OVERVIEW

This appendix provides a summary of the organization of all software modules and output files used to implement equipment malfunction diagnosis. The module names, contents, and general functionality are described. The system is comprised of approxiamtely 3,200 lines of C++ code. This code resides on the SUN 4 machine 'radon.berkeley.edu' in the ~bcam/src/diagnosis/lam directory.

File Name	Contents	Function Name(s)	Purpose
belief	Output directory for belief vs time data	none	not applicable
bpmd.cc	Member functions for bpm_node class	*~bpm_node*	Deletes an entire linked list of bpm_nodes
		iter	Iterates through a list of bpm_nodes
		append	Appends one bpm_node to the end of a list
		modify_bit_value	Ensures that the last bpm_node in a list is a 'theta' node
		my_length	Returns the number of bpm_nodes in a list
		my_nth	Returns the support slot for a given node in a list of bpm_nodes
		my_subst	Substitutes a support value into a given position in a list of bpm_nodes
		normalize	Performs BPMD normalization on a bpm_node list
	Friend functions for bpm_node class	*arb_bpmd_comb*	Performs arbitrary BPMD combination on two lists of bpm_nodes
		bel_pls_encript	Calculates final evidential intervals for a given bpm_node list
		bit_vec_equal	Returns the position of a given bit pattern in a bpm_node list
		my_bit_and	Performs the logical 'and' of the bit patterns of two different bpm_nodes
bpmd.h	Class definition for bpm_node class	none	not applicable
const.h	Symbolic constants for diagnosis software	none	not applicable

File Name	Contents	Function Name(s)	Purpose
evidence.cc	Member functions for evidence_node class	*~evidence_node*	Deletes an entire linked list of evidence_nodes
		iter	Iterates through a list of evidence_nodes
		append	Appends one evidence_node to the end of a list
		get_online_spc_info	Retrieves values for setpt and tol slots of on-line evidence list
	Friend functions for evidence_node class	*initialize_maintenance_evidence*	Initializes the maintenance evidence list
		initialize_online_evidence	Initializes the on-line evidence list
		initialize_inline_evidence	Initializes the in-line evidence list
evidence.h	Class definition for evidence_node class	none	not applicable
fault_set.cc	Member functions for fault_set_node class	*~fault_set_node*	Deletes an entire linked list of fault_set_nodes
		iter	Iterates through a list of fault_set_nodes
		append	Appends one fault_set_node to the end of a list
		clear_fault_bits	Clears the bits in every node of a fault-set list
		count_faults	Counts the total number of fault symbols in a fault-set list
		get_fault_bits	Converts a symbolic fault list to its integer representation

File Name	Contents	Function Name(s)	Purpose
	Friend functions for fault_set_node class	*char_to_int*	Returns the integer representation of a given ascii character
		initialize_fault_sets	Initializes the global fault-set list
		make_bpmd	Returns BPMD for given fault-set list, support and direction of constraint violation
		make_fault_set	Returns the fault-set list for a given piece of evidence
fault_set.h	Class definition for fault_set_node class	none	not applicable
faults.info	Component failure info for maintenance diagnosis	none	not applicable
file_io.cc	Member functions for recfile class	*recfile*	Creates a record file
		open	Opens a recfile using the standard C I/O function fopen()
		close	Closes a recfile using the standard C I/O function fclose()
		seek	Moves the recfile pointer to a given record
		rw	Reads/writes records in/out of a buffer at given starting record
		err_handler	Recfile error handling function
	Member functions for bpm_rec class	*display*	Displays a bpm_rec file
	Friend functions for bpm_rec class	*translate_lam1_flt*	Translates fault symbols to corresponding lam1 names

File Name	Contents	Function Name(s)	Purpose
file_io.h	Class definitions for recfile bpm_rec classes	none	not applicable
inline.cc	Main program to execute all 3 phases of lam1 diagnosis	none	not applicable
inline.rank	Fault ranking from in-line diagnosis	none	not applicable
inline_belief.cc	Functions necessary for in-line diagnosis	*inline_belgen*	Generates in-line support using reverse regression
inline_bpmd.fil	Record-oriented BPMD file for in-line diagnosis	none	not applicable
main_bpmd.fil	Record-oriented BPMD file for maintenance diagnosis	none	not applicable
maintenance.cc	Main program to execute lam1 maintenance diagnosis	none	not applicable
maintenance.rank	Fault ranking from maintenance diagnosis	none	not applicable
misc_diag.cc	Miscellaneous functions used for diagnosis	*bel_gen* *bits_to_char* *cusum_high*	Generates belief from 'squashing' function Converts sequence of bits to an ascii character Calculates cusum positive deviations using tabular V-mask

File Name	Contents	Function Name(s)	Purpose
		cusum_low	Calculates cusum negative deviations using tabular V-mask
		get_fault_mtf	Retrieves mtf from 'faults.info'
		get_fault_tsf	Retrieves tsf from 'faults.info'
		make_xgraph_files	Stores XGRAPH datasets from maintenance ranking in 'belief' directory
		read_bpmd_file	Reads support from record-oriented BPMD files into bpm_node lists
		store_bpmd_file	Stores a bpm_node list into a record-oriented BPMD file
		store_belief	Stores belief from each node of bpm_node list into appropriate file in 'belief' directory
		transfer_raw_data	Reads raw sensor data, transfers it to format for on-line diagnosis
online.cc	Program to execute on-line diagnosis	none	not applicable
online.data	Formatted file containing on-line sensor data	none	not applicable
online.rank	Fault ranking from on-line diagnosis	none	not applicable
online_bpmd.fil	Record-oriented BPMD file for on-line diagnosis	none	not applicable

APPENDIX 6.3

MANUAL PAGES FOR DIAGNOSIS SOFTWARE

This appendix provides the UNIX manual pages for the BCAM equipment malfunction diagnosis software. The commands described apply to the Lam Research Autoetch 490 plasma etcher in the Berkeley Microfabrication Laboratory. Commands are executable from the ~bcam/src/diagnosis/lam directory on radon.berkeley.edu.

NAME

inline - perform the maintenance, on-line and in-line phases of diagnosis on the Lam etcher

SYNOPSIS

inline *<recipe> <data> <measurements>*

DESCRIPTION

Inline performs malfunction diagnosis on the etcher after processing. In addition to running **maint**(1) and **online**(1), **inline** also performs in-line diagnosis using a summary of in-line etch measurements.

Recipe is the file which contains the etch recipe settings. *Data* is a file containing the real-time data produced by the LAMTALK equipment monitoring program. The *measurements* file contains a summary of the in-line etch measurements.

Inline produces an output file called 'inline.rank'. This file contains the component name, support and plausibility for each etcher fault. In addition, **inline** creates **xgraph**(1) datasets which contain diagnostic support versus time data. These datasets are stored according to component name in the ~bcam/src/diagnosis/lam/belief directory.

SEE ALSO

maint(1), online(1), xgraph(1)

AUTHOR

Gary S. May, University of California at Berkeley

Last change: May, 1991

NAME

maint - perform the maintenance phase of diagnosis on the Lam etcher

SYNOPSIS

maint

DESCRIPTION

Maint performs malfunction diagnosis on the etcher before processing. It reads component mean-time-to-failure and elapsed time since last failure data from a file called 'faults.info'. This file is generated and updated by running **pareto(1)** from ~gsm on argon.berkeley.edu. It must then be copied to ~bcam/src/diagnosis/lam/faults.info on radon.berkeley.edu.

Maint produces an output file called 'maintenance.rank'. This file contains the component name, support and plausibility for each etcher fault.

SEE ALSO

pareto(1)

AUTHOR

Gary S. May, University of California at Berkeley

Last change: May, 1991

NAME

online - perform the maintenance and on-line phases of diagnosis on the Lam etcher

SYNOPSIS

inline *<recipe> <data>*

DESCRIPTION

Online performs malfunction diagnosis on the etcher after processing. In addition to running **maint**(1), **inline** also performs on-line diagnosis using real-time sensor data.

Recipe is the file which contains the etch recipe settings. *Data* is a file containing the real-time data produced by the LAMTALK equipment monitoring program.

Online produces an output file called 'online.rank'. This file contains the component name, support and plausibility for each etcher fault. In addition, **online** creates **xgraph**(1) datasets which contain diagnostic support versus time data. These datasets are stored according to component name in the ~bcam/src/diagnosis/lam/belief directory.

SEE ALSO

maint(1), xgraph(1)

AUTHOR

Gary S. May, University of California at Berkeley

Last change: May, 1991

NAME

pareto - retrieve etcher component failure information from FAULTS

SYNOPSIS

pareto > faults.info

DESCRIPTION

Pareto retrieves etcher component mean-time-to-failure (mtf) and elapsed time since last failure (tsf) data from the FAULTS database. It directs the output into a file called 'faults.info'. This file contains each component's name, mtf and tsf. This information is necessary for maintenance diagnosis.

Pareto is executed from ~gsm on argon.berkeley.edu. The 'faults.info' file should be copied to ~bcam/src/diagnosis/lam/faults.info on radon.berkeley.edu.

SEE ALSO

maint(1)

AUTHORS

David Mudie and Gary S. May, University of California at Berkeley

Last change: May, 1991

CHAPTER 7

SYSTEM VERIFICATION AND CONCLUSIONS

7.1 Introduction

This chapter illustrates the use of the equipment diagnostic system for two typical malfunctions which occur in the operation of the Lam Research Autoetch 490 automated plasma etcher. Following a discussion of the diagnosis of these faults, a summary of the major results of this research and some suggestions for future work are presented.

7.2 Diagnostic Examples

Two typical diagnostic scenarios are discussed below. For each of these, the polysilicon etch rate, uniformity, and oxide and resist selectivity are monitored by standard Shewhart control charts with +/- 3-σ control limits [1]. In these charts, the etch outputs of the first nine baseline wafers are contrasted with 10 new wafers for which a simulated etcher malfunction has caused the process to shift. The word "simulated" here refers to the fact no actual malfunctions existed; the wafers were merely etched under conditions which would occur in the event of such a malfunction. The charts illustrate that even when process shifts do occur, they are not always easily discernible to an operator examining a control chart.

The nine baseline measurements come from wafers etched by the center point recipe of the modeling experiment described in Chapter 4. The process settings of this recipe are: pressure = 250 mtorr, RF power = 350 watts, CCl_4 flow = 125 sccm, He flow = 125 sccm, O_2 flow = 15 sccm, and electrode spacing = 1.5 cm. The shifts have been induced artificially by etching at slightly different recipes. The shifts were introduced by varying one of the controllable etch parameters from the center point recipe. These modifications simulate the behavior of the etcher under the conditions of the malfunction.

7.2.1 Miscalibrated Mass Flow Controller

One of the most common equipment malfunctions that occurs on the etcher is the miscalibration of a mass flow controller (MFC) [2]. When an MFC becomes miscalibrated, the flow of gas into the etcher's process chamber may be less than what is expected. This diagnostic scenario is illustrated in Figures 7.1 and 7.2.

Figure 7.1 shows the oxide selectivity control chart for 19 wafers. The first nine wafers were etched with the center point recipe, and the average selectivity for these wafers is 7.1. The 3-σ control limits of the control chart in the figure are estimated from the baseline wafers. For the next 10 wafers, a miscalibrated MFC is simulated by etching at a reduced CCl_4 flow (115 sccm). This results in a mean selectivity of only 5.5. This reduction in oxide selectivity due to lower CCl_4 flow is in agreement with trends predicted by the oxide selectivity model in § 4.4.3. The average etch rate (R_P), uniformity (U), oxide selectivity (S_{ox}) and resist selectivity (S_{ph}) for nominal and malfunctioning etcher operation are summarized in Table 7.1.

Table 7.1 - Summary of Etch Outputs for Miscalibrated MFC Example

Response	Nominal Operation	Malfunctioning Operation
R_P	4494 Å /min	4275 Å /min
U	8.9 %	8.6 %
S_{ox}	7.1	5.5
S_{ph}	2.5	2.6

Figure 7.2 shows the evidential support versus time plot during the processing of one wafer with a miscalibrated MFC. Here, the maintenance phase of diagnosis shows some indication of a fault. However, since gas flow sensors are directly connected to the circuitry which controls the MFC [3], a miscalibrated MFC is not detectable by these sensors. Therefore, on-line diagnosis cannot reveal this problem. Nevertheless, during in-line diagnosis, the system computes significant likelihood for a miscalibrated MFC fault.

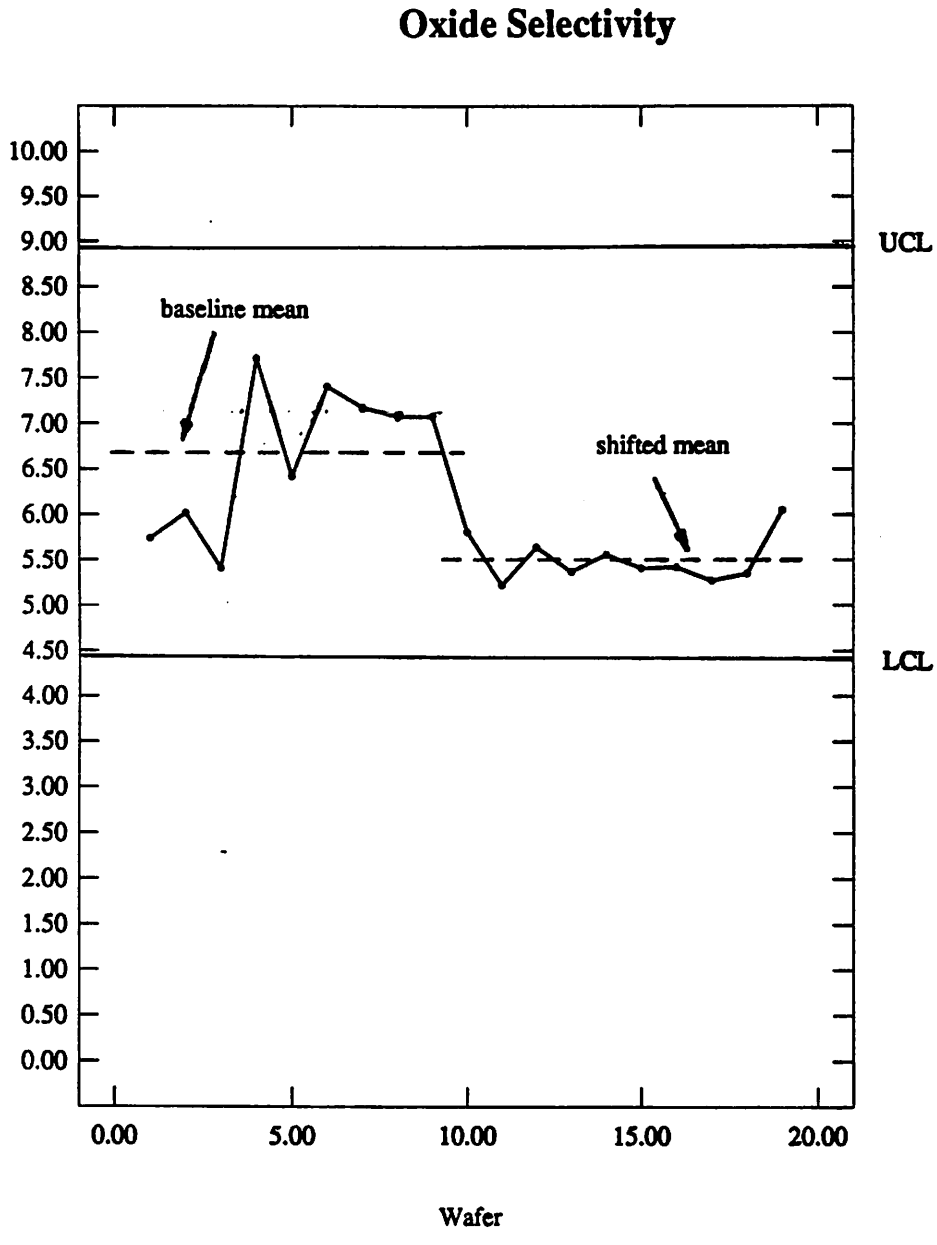


Figure 7.1 - Monitored oxide selectivity for miscalibrated MFC example.

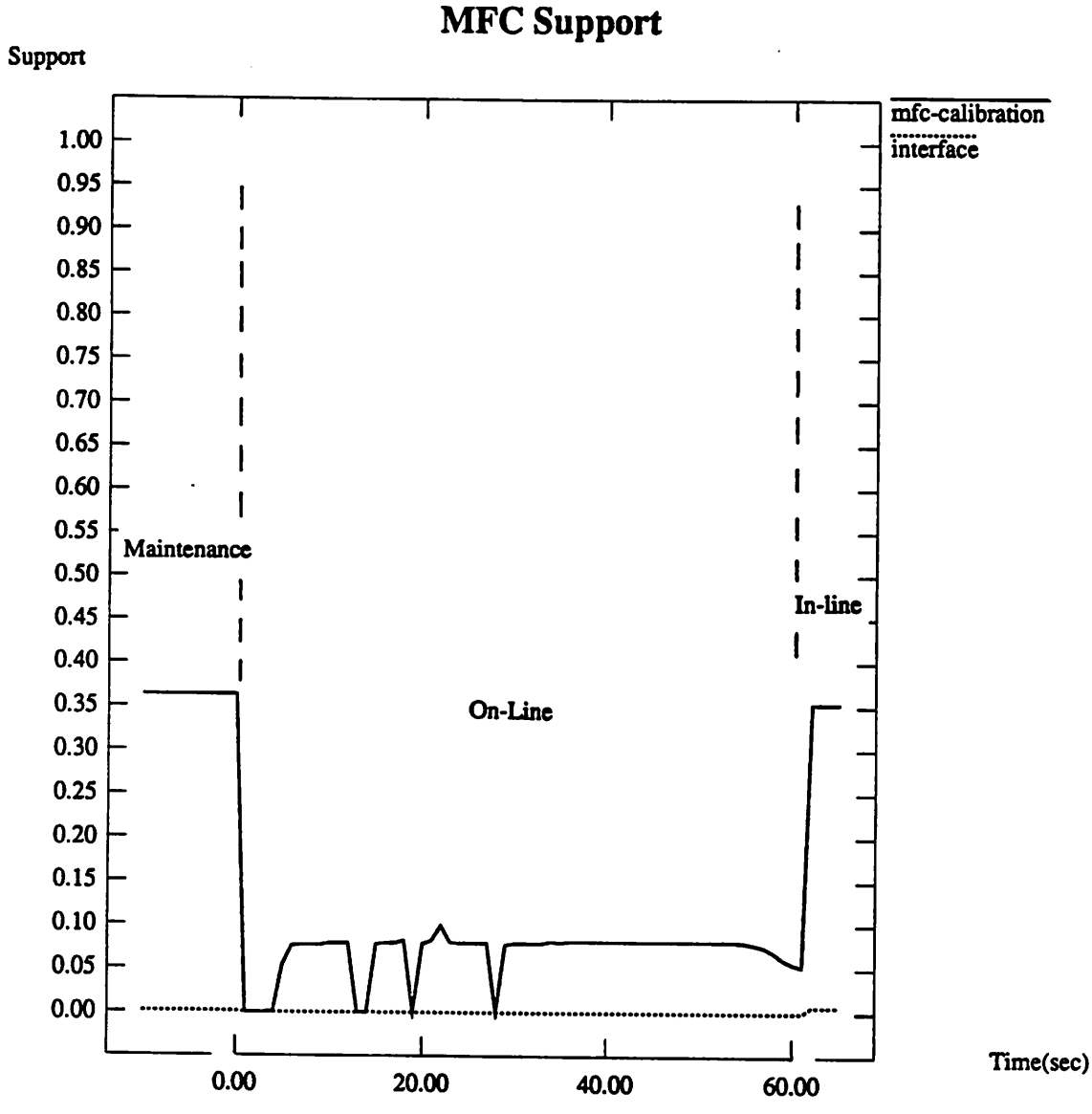


Figure 7.2 - Support versus time graph for MFC miscalibration.

Recall from the Fault Propagation diagrams in Appendix 3.2 that an MFC miscalibration is occasionally the result of a malfunctioning microprocessor. The microprocessor is a part of the etcher's interface electronics system. Consequently, in Figure 7.2, the support for the MFC fault is contrasted with that of a fault in the interface electronics. However, in this case, a fault in the electronics system is not discernible.

7.2.2 Phase/Magnitude Detector Problem

Another fault which may occur in the plasma etcher involves the phase/magnitude detector system which matches the chamber impedance to the RF generator output. This matching is necessary to achieve maximum power transfer [3]. Occasionally, however, the timing mechanism in the phase/magnitude detector malfunctions and causes a power loss [4]. This loss of power can significantly reduce etch rate. This situation is depicted in Figures 7.3 and 7.4.

The first nine wafers in Figure 7.3 are etched with the same center point recipe as in the previous example. For these, the average etch rate is once again 4494 Å /min. Once again, the control limits of the control chart are estimated from the baseline wafers. For the next 10 wafers, the phase/magnitude detector problem is simulated by reducing the forward RF power to 340 watts, which results in a decrease in mean etch rate to 4027 Å /min. This reduction is also in agreement with the model in § 4.4.1. The remaining etch outputs for this example are summarized in Table 7.2.

Table 7.2 - Summary of Etch Outputs for Phase/Magnitude Detector Example

Response	Nominal Operation	Malfunctioning Operation
R_P	4494 Å /min	4027 Å /min
U	8.9 %	4.4 %
S_{ox}	7.1	5.3
S_{ph}	2.5	2.6

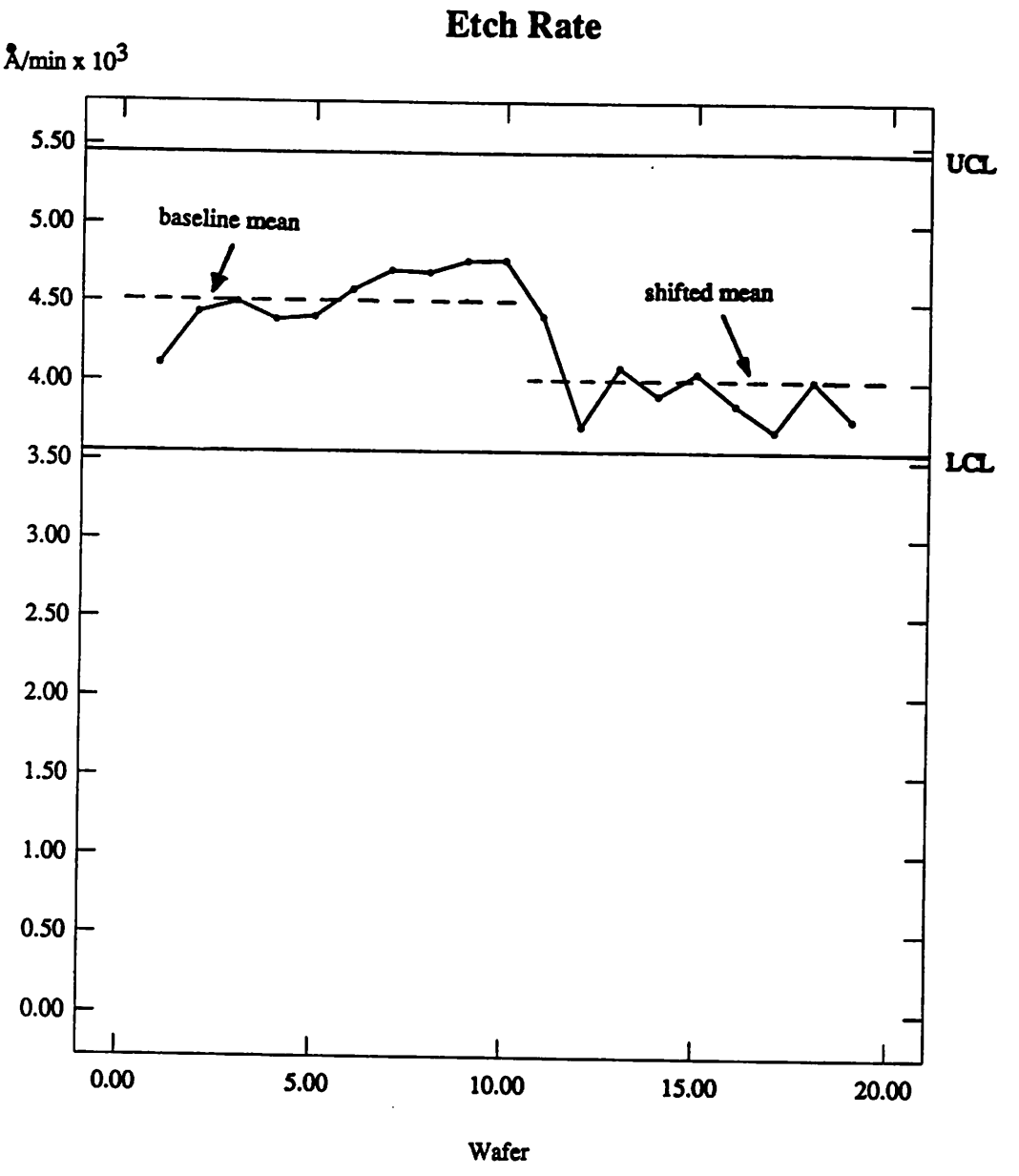


Figure 7.3 - Monitored etch rate for faulty phase/magnitude detector example.

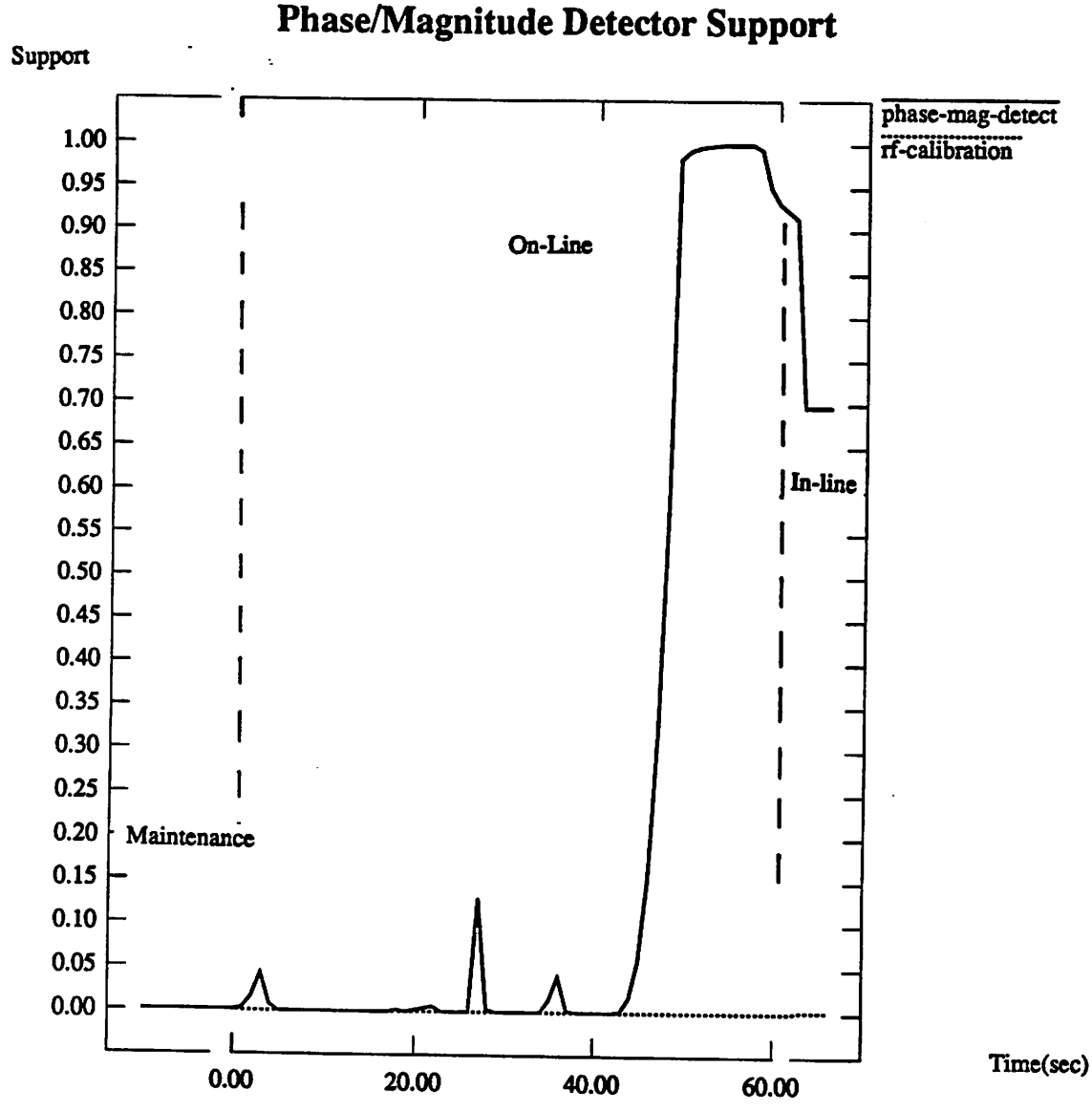


Figure 7.4 - Support versus time graph for phase/magnitude detector problem.

Figure 7.4 shows the support versus time plot from a misprocessed wafer under these conditions. Since the phase/magnitude detector fails very infrequently [2], the maintenance phase of diagnosis shows negligible indication of a fault. During on-line diagnosis, however, the support for this fault increases dramatically as the wafer is processed, and the system correctly identifies the phase/magnitude detector as a probable malfunction. This diagnosis is even further corroborated by the in-line phase.

The evidential support for the phase/magnitude detector problem may be compared to that of the RF calibration fault which is also depicted in Figure 7.4. Although a miscalibrated RF power system is also a possible fault when power-related malfunctions are observed (see Appendix 3.2), this graph shows that such miscalibration is unlikely in this case.

7.3 Conclusions

This dissertation has presented a general methodology for the automated malfunction diagnosis of semiconductor manufacturing equipment. The approach utilized combines the best aspects of algorithmic and knowledge-based methods. By employing Dempster-Shafer evidential reasoning [5] to infer the causes of failures, and by using evidence generated before, during and after equipment operation, malfunctions may be identified before significant misprocessing has occurred.

The development of this diagnostic system required a thorough understanding of equipment operation under optimal conditions as well as cognizance of the effect of individual malfunctions on nominal operation. Consequently, a qualitative model of both the interaction between equipment components and the propagation of equipment faults through the system has been developed for the piece of equipment which served as the application vehicle for the diagnosis prototype: the Lam Research Autoetch 490 plasma etcher [3].

In addition, the use of the Dempster-Shafer methodology necessitated comprehensive experimentation to obtain empirical models for the behavior of the Lam etcher. Thus, a series of experiments have been conducted to characterize the etch rate, uniformity, selectivity, and anisotropy of a polysilicon etch process using response surface models. These models have been shown to describe the critical etch outputs very precisely [6]. Moreover, the models provide quick analytical solutions which are essential for diagnosis.

Techniques were also developed to derive important diagnostic information from all three primary sources of evidence: equipment maintenance history, real-time sensor data, and the empirical equipment models. Each source of evidence corresponds to one phase of diagnosis, and each diagnostic phase is accompanied by a unique method of evidential support generation. For the maintenance phase, support is generated by comparing the elapsed time since a component has failed to that component's mean-time-to-failure. Both of these quantities are obtained from an equipment database [7]. During on-line diagnosis, support is derived by monitoring sensor data [8] and filtering correlation effects using the cumulative sum of measurement residuals [1]. Finally, for in-line diagnosis, an innovative technique has been developed to obtain evidential plausibility for equipment failures from the empirical models of equipment behavior [9]. In so doing, the this technique provides an important link between statistical modeling and knowledge-based diagnostic systems.

However, the Dempster-Shafer approach to diagnosis is not totally without limitations. Like many Bayesian schemes designed for large systems [10-11], Dempster-Shafer theory assumes independence of evidence sources. This is probably erroneous in integrated circuit fabrication equipment since sensor data is often correlated in time [12]. Although the effect of the independence assumption can be somewhat lessened by applying the Dempster-Shafer model to multiple fault groups in the manner described in § 2.5, accurate diagnosis with this technique requires an exhaustive mapping of evidence sources to particular faults or fault groups. Such a

mapping is not easily derived for complex systems.

The diagnostic methodology described in this thesis serves as one module of the Berkeley Computer-Aided Manufacturing (BCAM) architecture. Among the other capabilities of the BCAM system are: statistical process control (SPC), recipe generation, real-time equipment monitoring, modeling and simulation of equipment and processes, and automated maintenance record-keeping. In this framework, the diagnostic module plays a unique role in linking the various other applications.

Diagnosis is initiated when alarms are generated by SPC monitoring schemes. Also, in order for real-time diagnosis to take place, the link between the diagnostic module and the equipment monitoring software is critical. Furthermore, the maintenance phase of diagnosis depends heavily upon information obtained from the automated record-keeping system [7]. Finally, the empirical models of equipment behavior necessary for in-line diagnosis are equally useful for recipe generation [13] and manufacturing-based technology CAD simulation [14].

7.4 Future Work

7.4.1 Short Term

In the short term, only one task is of primary importance for plasma etch diagnosis. A more direct link between the monitoring software and the diagnostic module must be established. Such a link would allow the real-time analysis of sensor data during on-line diagnosis. At present, on-line diagnosis is performed by reading files containing recently transmitted sensor data (see § 6.3.2). Since executing the ONLINE program on a Sun 4 computer currently takes less than 30 seconds to process data collected during a 60 second process run, it is not unreasonable to assume that once such a direct link is established, real-time etch diagnosis may take place.

Moreover, aside from enabling real-time diagnosis, the link to the monitoring software

would permit the continuous updating of the standard deviations of all sensor inputs. These standard deviations are necessary for the implementation of the CUSUM support generation scheme described in § 5.3.3. Currently, these deviations are "hard-wired" into the "get_online_spc_info" function (refer to § 6.2.2).

7.4.2 Long Term

One way to lessen the impact of the limitations of the Dempster-Shafer approach outlined above involves optimizing the values of the various adjustable parameters used to generate and distribute numerical fault belief. These parameters include system uncertainty, the "sharpness" factor of equation (5.1), and the weighting system used to assign portions of fault belief to individual faults or fault groups (see § 5.4 and 6.2.3). Such an optimization requires extensive use of experimentation and simulation of known malfunctions in a manner similar to the examples presented in this chapter. The result of this effort would provide greater assurance that the diagnostic system is sufficiently robust to arrive at accurate conclusions over a wide range of failure scenarios.

The overall accuracy of the diagnostic system would also be greatly enhanced through the use of additional real-time sensors. The plasma etcher to which the system has been applied in this dissertation has a limited number sensors which provide useful data (see § 4.2). The lack of availability of sufficient sensors is a significant hindrance to evidence gathering. Improvement of sensor capabilities for the plasma etcher remains a critical topic of ongoing research for both universities and equipment manufacturers, and the application of this diagnostic methodology to an etcher with more abundant and more advanced sensors (such as the Lam Rainbow etcher [12]) would provide a better measure of the potential usefulness of the system.

Finally, although the techniques developed in this thesis have thus far been applied to fault identification in a single-wafer plasma etching system, the overall diagnostic approach is general

enough to be useful in other equipment applications. In order to alleviate some of the difficulties discussed above, further research should be initiated with the goal of applying this methodology to other fabrication processes. A few potential candidates include ion implantation and molecular beam epitaxy (MBE). Moreover, through the use of *functional decomposition* techniques [15], this diagnostic approach might also be useful in the development of state-of-the-art "cluster tools" and other multi-chamber, single-vacuum apparatus.

References for Chapter 7

[1] D. C. Montgomery, *Introduction to Statistical Quality Control*, 2nd Edition, New York: Wiley, 1989.

[2] R. Norman, *Private Communication*, June, 1990.

[3] *Autoetch Plasma Etch System Operation and Maintenance Manual*, Lam Research Corporation, March, 1985.

[4] A. Miller, *Private Communication*, February, 1991.

[5] G. Shafer, "A Mathematical Theory of Evidence," *Princeton University Press*, 1976

[6] G. S. May, J. Huang, and C. J. Spanos, "Statistical Experimental Design in Plasma Etch Modeling," *IEEE Transactions on Semiconductor Manufacturing*, vol. 4, no. 2, May, 1991.

[7] D. Mudie and N. H. Chang, "FAULTS: An Equipment Maintenance and Repair System Using a Relational Database," *Proceedings of the 1990 IEEE/CHMT International Electronics Manufacturing Technology Symposium*, October, 1990.

[8] N. H. Chang, "Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," *UC-Berkeley Electronics Research Laboratory Memorandum No. UCB/ERL M90/61*, May, 1990.

[9] C. J. Spanos and G. S. May, "Using Regression Equations for Model-Based Diagnosis in the Berkeley Computer-Aided Manufacturing System," *Workshop on Intelligent Diagnostic and Control Systems for Manufacturing*, Boston, July, 1990.

[10] R. O. Duda, J. G. Gasching, and P. E. Hart, "Model Design in the Prospector Consultant System for Mineral Exploration," *Expert Systems in the Microelectronic Age*, Edinburgh: Edinburgh University Press, 1980.

[11] D. S. Vaughan, B. M. Perrin, R. M. Yadrick, and P. D. Holden, "Comparing Expert Systems Built Using Different Uncertain Inference Systems," *Proceedings of the Fifth Annual AAAI Workshop on Uncertainty and AI*, August, 1989.

[12] H. Guo, C. J. Spanos, and A. Miller, "Real Time Statistical Process Control for Plasma Etching," *Proceedings of the 1991 International Semiconductor Manufacturing Science Symposium*, May, 1991.

[13] K. K. Lin and C. J. Spanos, "Statistical Equipment Modeling for VLSI Manufacturing : An Application for LPCVD," *IEEE Transactions on Semiconductor Manufacturing*, vol. 3, no. 4, November, 1990.

[14] T. L. Luan, G. S. May, H. C. Liu, and C. J. Spanos, "Using Equipment Models within a Technology CAD Framework," submitted to the *1991 International Conference on Computer-Aided Design.*

[15] F. E. Finch and M. A. Kramer, "Narrowing Diagnostic Focus Using Functional Decomposition," *AIChE Journal*, vol. 34, no. 1, January, 1988.