PIX (PROGRAM USER INTERFACE IN X)

by

Suet M. Chiu

Memorandum No. UCB/ERL M91/64

29 June 1991

PIX (PROGRAM USER INTERFACE IN X)

by

Suet M. Chiu

Memorandum No. UCB/ERL M91/64

29 June 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PIX (PROGRAM USER INTERFACE IN X)

by

Suet M. Chiu

Memorandum No. UCB/ERL M91/64

29 June 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PIX

(Program user Interface in X)

Suet M. Chiu

Electronics Research Laboratory,
Department of Electrical Engineering & Computer Sciences,
University of California, Berkeley, CA 94720

ABSTRACT

PIX (Program user Interface in X) is a graphical user interface designed to facilitate the communication between the user and simulation programs such as PARMEX and SAMPLE. PIX features pull-down menus and dialog boxes based on the X Toolkit Athena Widget Set.

PIX has been developed to run with PARMEX for automatically modeling acid-hardened resist from dissolution measurement. It allows the user to view the data while using mouse driven commands interactively to perform data reduction and curve fitting.

In this report, the internal structure of PIX is described in detail. Issues regarding display interface and application interface are discussed. Information is provided to assist future extensions for incorporating new simulation tools. PIX-PARMEX has been used to model the i-line resist KTI-895i and the acid-hardened resist, Shipley XP-8843. Examples are given to show how PIX helps PARMEX users to customize the modeling process for the two resists.

Acknowledgements

I would like to thank Professor Andrew R. Neureuther for his valuable guidance and encouragement. His time and patience throughout this project are greatly appreciated. I would also like to thank Professor William G. Oldham for his valuable comments and his reading of the paper.

I wish to thank my colleagues for their support and advice. I thank Richard Ferguson, Nelson Tam and John Hutchinson for their valuable PIX feedback. To the friends who have encouraged and guided me throughout my stay at Berkeley, I thank Robert Wang, Charmine Tung, Annie Lai, Anantha Chandrakasan and Chi-woo Chan for their friendship and company during lunches and dinners.

Finally, to my parents, I am deeply thankful for their loving support.

The financial support from SRC is gratefully acknowledged.

Table of Contents

Chapter 1

Introduction

1.1 History and Background

Before personal computers, most programs were custom-built and required special training to use. Every program had its own commands and syntax. It took users a long time to differentiate which command to use or what data operands a command requires for a particular program. They had to memorize the syntax or look up the command in a thick manual every time they tried to run the program. The computer time and power were more precious than human time. It was easier and cheaper to train users than to do the extra work to build software in a better way that helped the users to eliminate the painful learning process.

Beginning in the early 1980s, the personal computer software market was booming. Software companies started to realize that a good user interface increased the competitiveness of their softwares in the market. The emphasis until the late 1980s was still on optimizing two scarce hardware resources, computer time and memory. Program efficiency was the highest goal. With today's low hardware costs and increasingly powerful graphics-oriented personal-computing environments, however, we can afford to optimize user efficiency rather than just computer efficiency. The popularity of engineering workstations with great computing and graphics power makes building graphical user interface an easier, cheaper task than before.

The quality of Graphical User Interface (GUI) often determines whether the software is successful. For example, the Macintosh desktop user interface with its windows, icons and pull-down menus, is popular because it is easy to learn and requires little typing skill. Most users are not computer programmers and do not like the old-style, hard-to-learn, keyboard-oriented interfaces. A good GUI must create a smooth communication environment between the user and the program. Being sensitive to users' desire for easy-to-learn, a GUI contributes a great deal to a program's acceptance and success.

1.2 PIX Overview and Features

With the user-friendly concept as the goal, **PIX** (*Program user Interface in X*) is a graphical user interface program designed to facilitate the communication between the user and simulation programs such as PARMEX [1] and SAMPLE [2].

PIX was developed on X Windows in UNIX. It uses the X Toolkit Intrinsics and the X Toolkit Athena Widget Set [3]. PIX has been designed to be easily extensible and modifiable. It has been written in a modular fashion so that new programs and new program commands can be added in a straightforward fashion.

A *display* interface controls the interaction of PIX with the X window system. It manages the visual appearance (*look and feel*) of every program supported by PIX. The main goal is to present a consistent "look and feel" for all the programs to the user. Consistency in appearance and structures help the user develop his or her own way of controlling the application and thus is an important quality for GUI.

The current version of PIX supports PARMEX, a resist parameter extraction program, and Drawplot, a two-dimensional plotting program. PARMEX also runs with SAMPLE, a process simulator, to obtain resist exposure information. The block diagram in Figure 1.1 gives a global picture of the PIX system. PIX presents a command menu of the simulation program to the user for selection. After the user has selected a command, PIX makes up the command line and sends it to the background program for processing. As soon as the execution is complete, the results from the program are sent back to PIX and displayed in the PIX window. Then, PIX is ready for the next command selection. This simple communication protocol between the user and the program through PIX forms an important skeleton for the interface program. It has been termed as the PIX *application* interface.

PIX also communicates with Drawplot, the plotting program, as shown in Figure 1.1. Graphical data are plotted in separate PIX windows. The plot window can be resized, printed and iconized for later display. PIX can automatically open multiple plot windows and allow users to examine them on the screen simultaneously. Figure 1.2 shows the "look and feel" of PIX with a separate graphical plot window.

This report discusses the methodology for the design of PIX. Its features and data structures are described in details. Chapter 2 describes the PIX display structures based on the Athena Widget Set of the X window system. Extra features such as plotting capabilities, dialog selection objects and interactive/batch mode are also described in this chapter. Chapter 3 discusses the application interface with PARMEX. Detailed implementation techniques are provided to assist developing additional interfaces with other tools. Chapter 4 presents examples which shows how PIX-PARMEX was used to model the i-line exposed resist, KTI-895i, and the acid-hardened resist, Shipley XP-

8843. Pictures with descriptions are provided to show the results of various stages in the execution of PIX-PARMEX, and to demonstrate how PIX-PARMEX performs data reduction and model fitting. Chapter 5 gives an overview of future directions for PIX. PIX can be extended to support both 2D and 3D SAMPLE. The tradeoffs and tips are described fully.

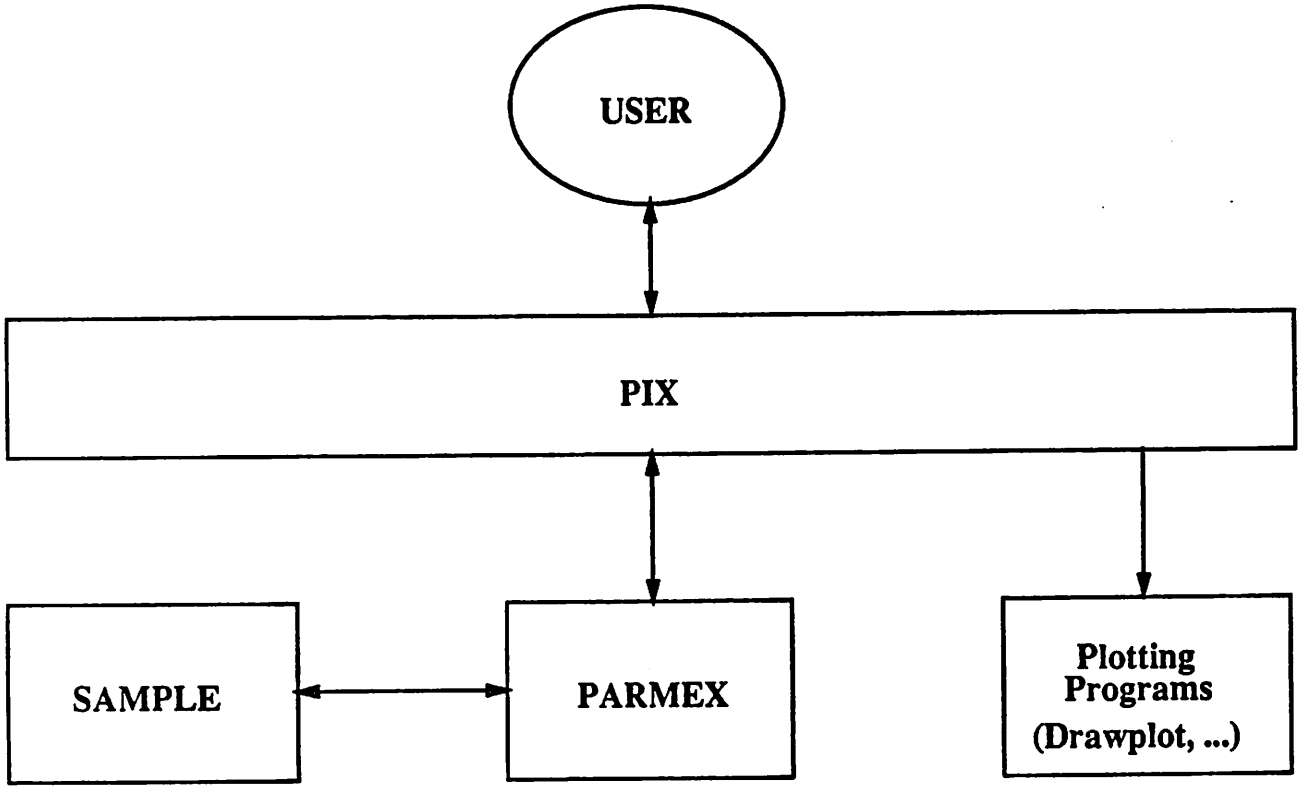

Figure 1.1 PIX System Block Diagram

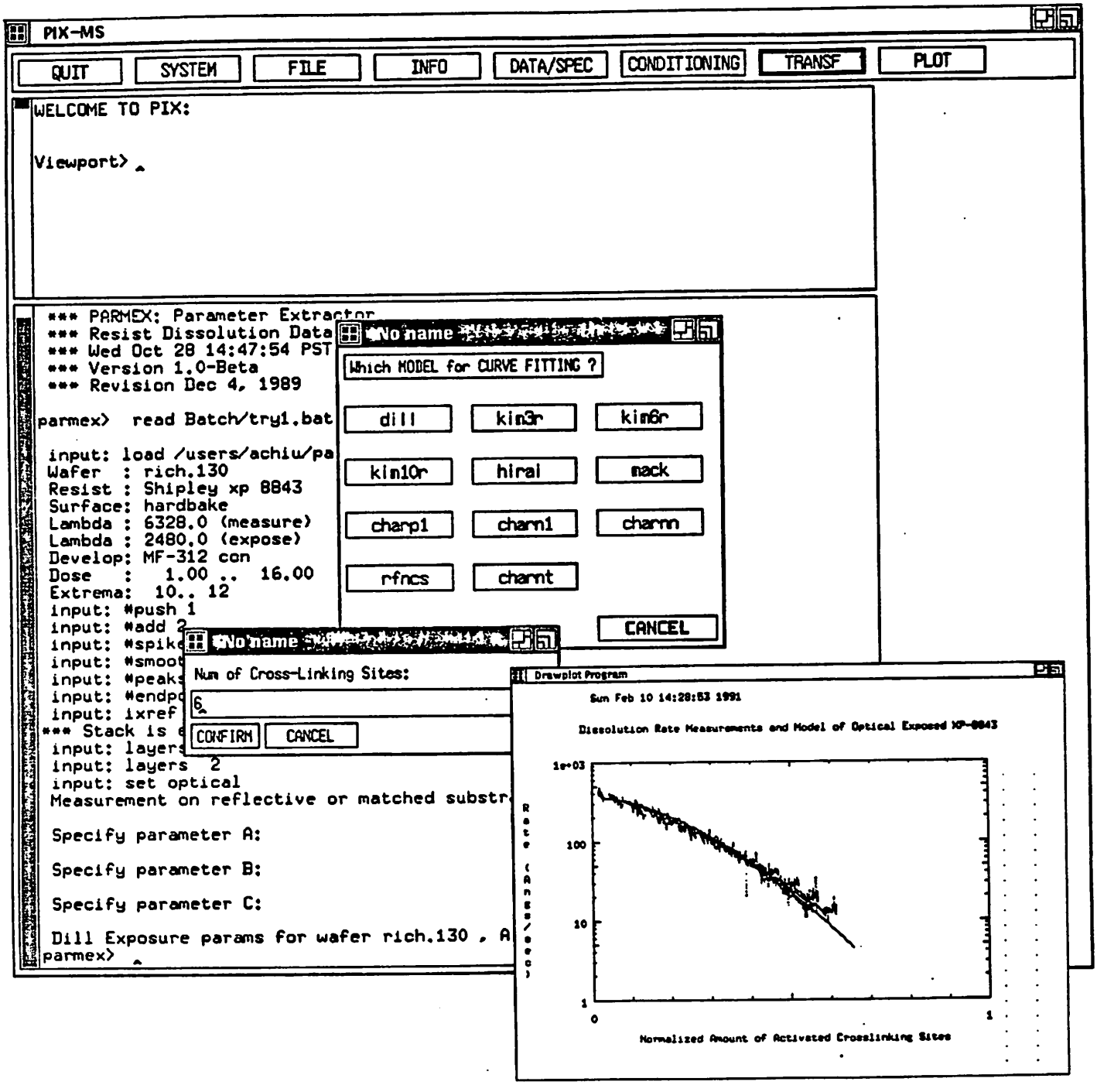

Figure 1.2 The "look and feel" of PIX

Chapter 2

Display Interface with Athena Widget Set

PIX was developed on X Windows in UNIX. Its display structure is based on the X Toolkit Intrinsics and the X Toolkit Athena Widget Set. The basic components of PIX include the menu selection panel featuring pull-down menus and two console windows.

2.1 Definition and background

The Intrinsics and the Athena Widget Set make up the X Toolkit. The Intrinsics is a library package layered on top of Xlib [4]. Xlib functions are basic functional calls to the X Server which interacts with the display hardware. Figure 2.1 shows the X Window programming hierarchy. The functions and structures in the Intrinsics extend the basic programming abstractions provided by the X Window system. With the mechanisms for intercomponent and intracomponent interactions, the Intrinsics provides the next layer of functionality from which the Athena Widget Set, HP Widget Set, MOTIF [5], etc., are built.

Application programs such as SIMPL-DIX [6], Drawplot, Pdraw [7] are built on top of Xlib only. On the other hand, PIX is built on top of the X Toolkit Athena Widget Set as shown in Figure 2.1. Both the Athena widgets and Xlib are in the public domain and can be obtained from Massachusetts Institute of Technology.

There are tradeoffs in using Athena Widget Set. Because the Intrinsics mask implementation details from the widget and interface application program (PIX), the current Athena widgets are fully extensible and can be replaced by higher performance and functionality widgets such as MOTIF with no major change to the program. This

makes upgrade fairly easy. However, different workstations may support different versions of X Intrinsics. For example, the functions of the Text Widget of Athena Widget Set Version 3 have been renamed from *XtText_function()* to *XawText_function()* in Version 4. As a result, the user interface program using this widget needs to be updated for the change. In general, programs based on Xlib are compatible for most of the machines whereas those based on the widget set might be not. Similar to the Xlib upgrade from version 10 to version 11, this problem will be eliminated when the new version becomes universal and is used across all the workstations.

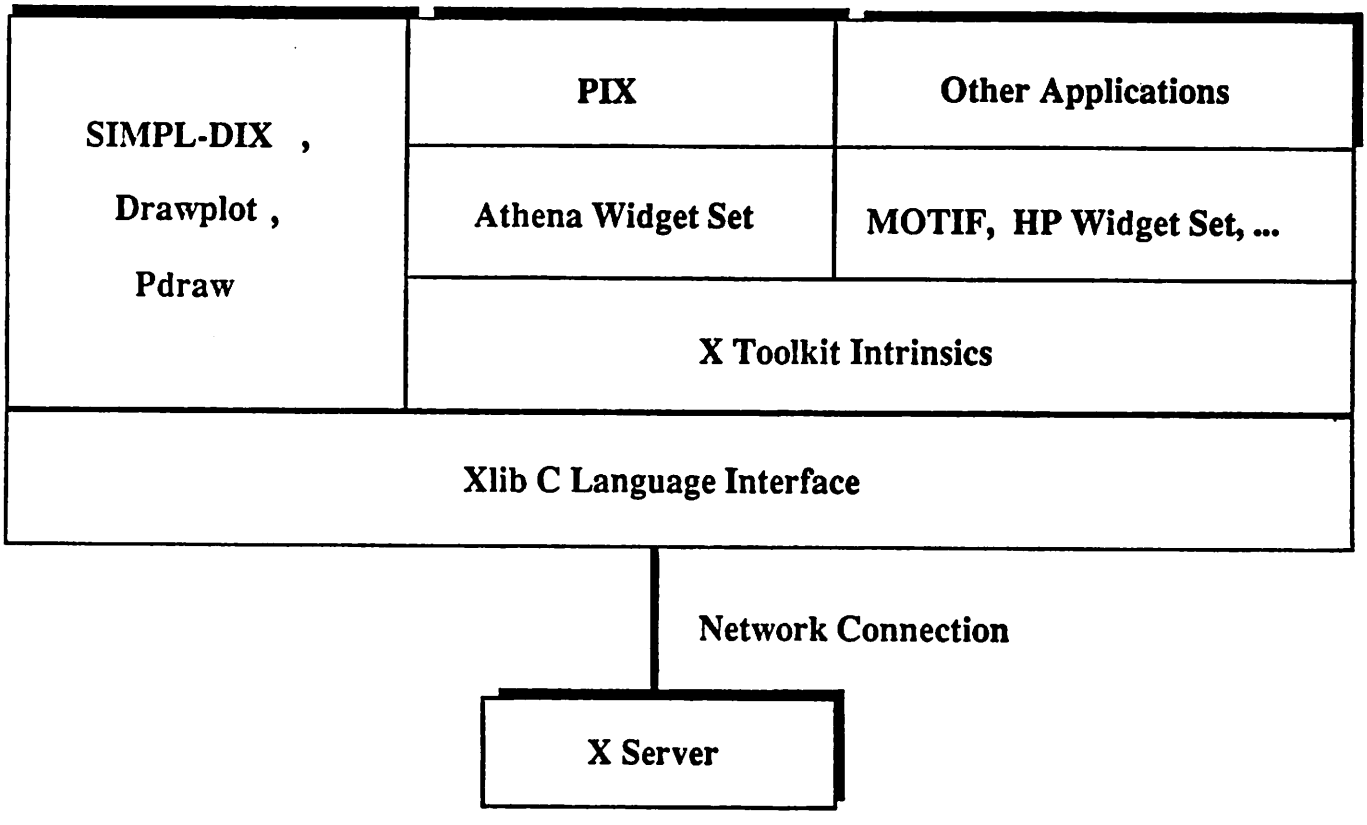

Figure 2.1 X Window Programming Environment

TERMINOLOGY

Application programmer

A programmer who uses the X Toolkit to produce an application user interface.

Button

A box-shaped object that is sensitive to the motion of the mouse cursor.

Child

A widget that is contained within another "parent" widget.

Client

A function that uses a widget in an application or for composing other widgets.

A Callback Routine

A procedure with a widget that is called under certain conditions. For example, the Command widget has a callback routine to notify clients when the button has been pushed and released.

Object

A software data abstraction consisting of private data and functions that operate on the private data. Users of the abstraction can interact with the object only through calls to the object's public functions. In the X Toolkit, some of the object's public functions are called directly by the application, while others are called indirectly when the application calls the common Intrinsics functions.

User

A person interacting with a workstation.

Widget

An object providing a user-interface abstraction (for example, a Dialog widget).

Window

An area on the display screen associated with an application.

Viewport

A pane of a window into which an application maps output.

Command widget

A rectangular button that contains a text or pixmap label. When a button is pressed and released, the button is selected, and the application's callback routine is invoked.

Dialog widget

A widget which implements a commonly used interaction semantic to prompt for auxiliary input from a user. For example, a Dialog Widget is used when an application requires a small piece of information, such as a file name, from the user.

Text widget

A window that provides a way for an application to display one or more lines of text. It consists of a vertical Scrollbar in the Text window which lets the user scroll through the displayed text.

Form widget

A widget which can contain an arbitrary number of children or subwidgets. It provides geometry management for its children, which allows individual control of the position of each child.

2.2 Window Management

The PIX window is divided into four main components: group panel, shell viewport, console viewport and dialog selection panel. Figure 2.2a shows the relative positions of these components.

The group panel is located at the top of the PIX display. It is used by moving the mouse cursor onto small command boxes that are called group buttons. The group buttons arranged in a row in the panel, represent various command categories. The category names such as "QUIT", "SYSTEM" and "FILE" are displayed on the buttons.

Each group button in the panel automatically displays a menu when selected. Each menu consists of a group of command buttons belonging to that category. The menu buttons which are formatted vertically represent commands for PIX and the simulation program.

The number of group buttons in the group panel and command buttons in each menu depends on the simulation program. When the simulation program is invoked, the group buttons for the command categories of the program are created and displayed on the screen. After the user has clicked on the mouse with the cursor in the chosen group button, the group button is high-lighted. At the same time, the menu is popped up and displayed below the group button.

As the user selects one of the command buttons within the menu, the menu disappears, and a dialog selection panel is displayed to fetch any further information before the command line is made up. The dialog selection panel contains different dialog objects which are described in section 2.4.

The shell and console viewports are text widgets and do not respond to the keyboard or the mouse. Both of them have vertical scroll bars to roll up and down the messages displayed inside the viewports. The shell viewport only displays the results of shell commands selected by the user. These shell commands for the operating system are used to access directory information. For example, *ls* is a shell command to display all the names of the files in the current working directory of the computer. On the other hand, the console viewport displays the messages and results of the simulation program. After the user has selected a command through the menu panel, the command is executed. At the same time, the command is displayed in the console viewport to let the user see what command he or she has selected. As soon as the execution is completed, all the results and messages from the simulation program are displayed in the console window.

2.3 Menu Structure

PIX is designed to display a consistent *look and feel* for users. The first three group buttons: QUIT, SYSTEM and FILE, stay the same for every simulation program. The QUIT group consists of the quit-session and quit-analysis commands. The SYSTEM group consists of the shell commands for the operating system to help users access directory information. The FILE group contains commands to manipulate files such as *edit*, *load* and *save*. This arrangement of menu follows the Macintosh convention.

Figure 2.3a presents the top group menu before running any simulation program. Figure 2.3b shows the group menu after PARMEX is invoked. Figure 2.3c shows the plot group menu. Notice that the QUIT, SYSTEM and FILE groups are shared by all

three running modes.

Beside running simulation, PIX can also call out different plotting programs for SAMPLE plot files and PARMEX plot files. If the FILE group in the top group menu is selected, the simulation group will be changed to the PLOT group which consists of the choices of different plotting programs. As a result, graphical data can be plotted in separate PIX windows in a non-simulation environment. The creation of the plotting mode in PIX integrates the plotting programs together and hides the plot command-line syntax from the user.

2.4 Dialog Selection Objects

PIX supports three major kinds of dialog selection objects to obtain data for the needs of different command formats. The objects are contained in the dialog selection panel which is popped up when the command selected requires additional choices or data. Each object used the command widget for selection buttons and the dialog widget for queries. The first kind of dialog object presents button selections only, similar to the menu panel, but in a two dimensional order. Figure 2.4a shows an example of the selection of fitting models from all the available models in PARMEX. The second dialog object only presents dialog queries which obtain data from the user through the keyboard as shown in Figure 2.4b. The third object is a combination of the above two types. It consists of one query dialog and many button selections. An example of this object is shown in Figure 2.4c. The three dialog objects give users a consistent *look and feel* and will help users develop their own ways of controlling the program.

The three dialog objects are written in a general and flexible fashion. The number of selections, the number of dialog queries, selection labels, dialog labels and questions are not fixed. They are passed as parameters into the object routines. As a result, the objects give the programmer lots of freedom in designing dialog boxes to best fit the command formats in different simulation programs. To add a new command, the interface programmer needs only to add the command to the appropriate group, select the respective dialog object routine to be used, and write a callback routine for that new command. No new dialog object is needed.

2.5 Interactive/Batch Mode

All the command selections during a PIX session can be saved and rerun as a batch job. After the execution of the batch job, PIX will resume an interactive mode and wait for the next command. PIX users do not need to remember the commands and sequences of commands they have selected as the command lines are recorded in a batch file for future use. This flexible feature greatly helps the user to customize and automate the simulation process.

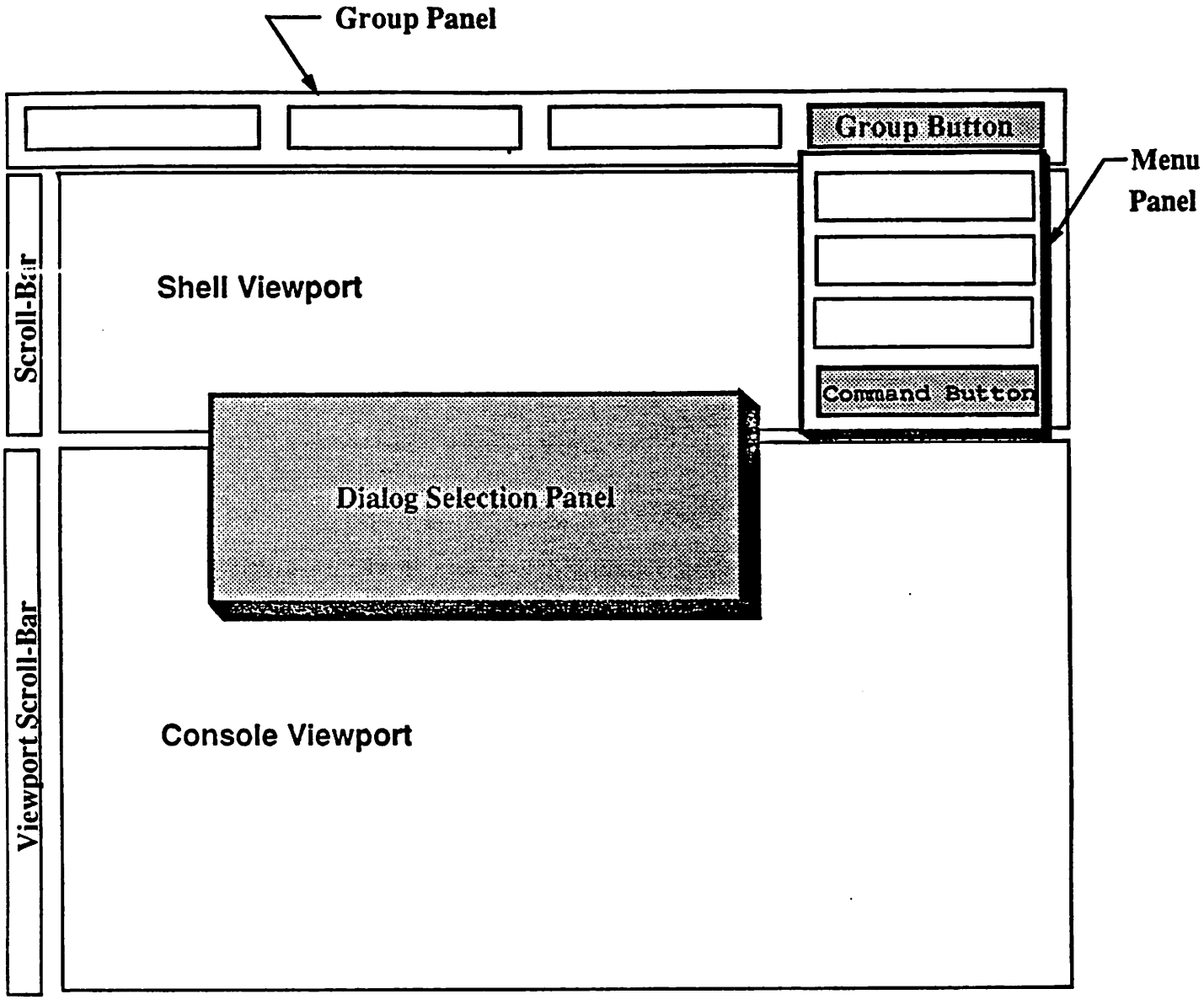

Figure 2.2a PIX Display Structure

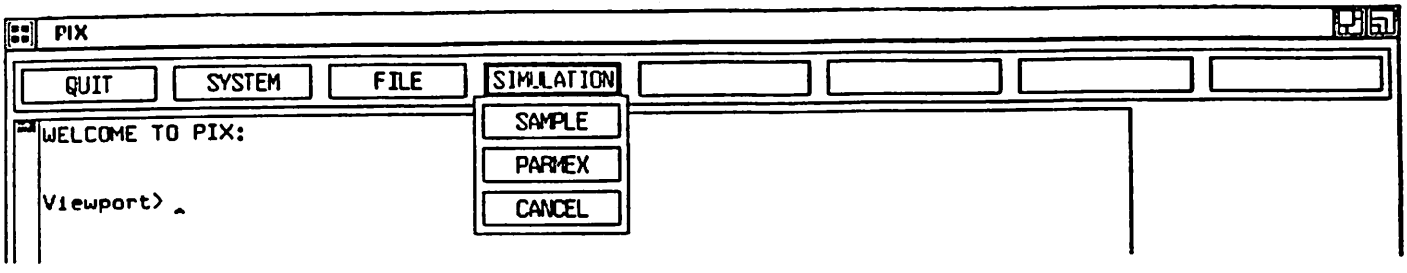

Figure 2.3a Top Group Menu

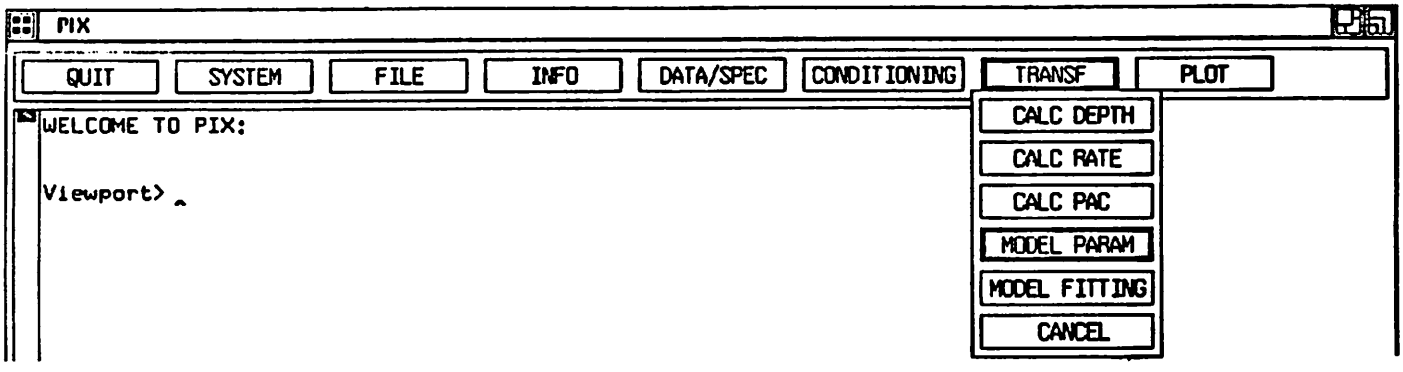

Figure 2.3b PARMEX Group Menu

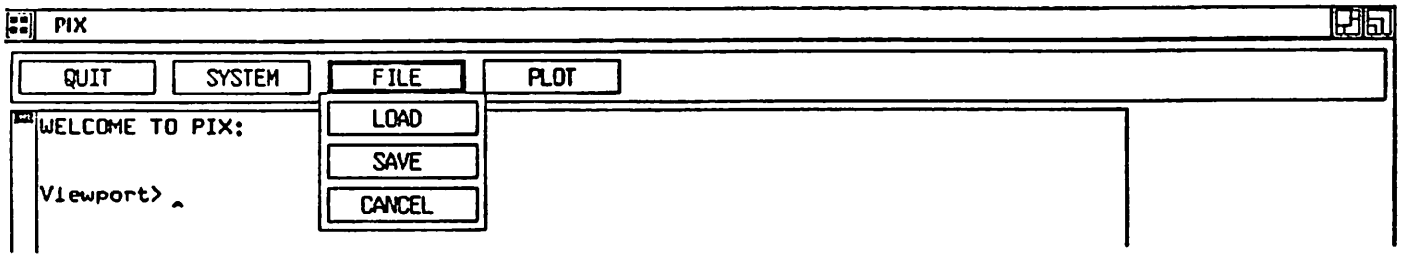

Figure 2.3c Plot Group Menu

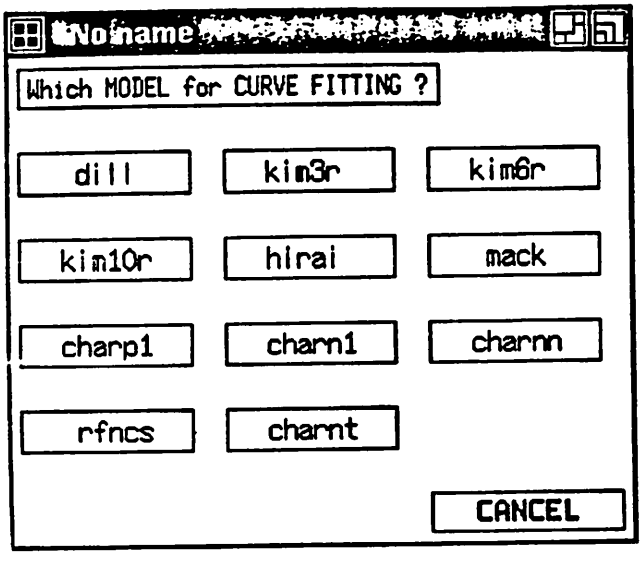

Figure 2.4a Dialog Selction Object I

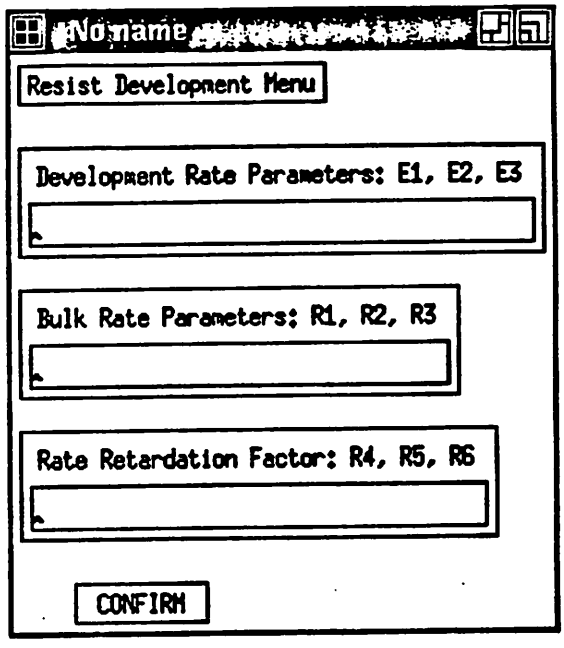

Figure 2.4b Dialog Selection Object II

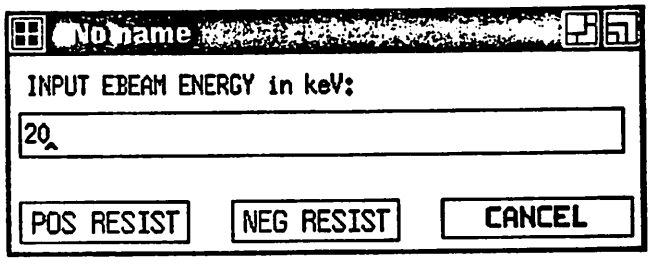

Figure 2.4c Dialog Selection Object III

Chapter 3

Application Interface with PARMEX

3.1 Introduction

The main function of a user interface program is for communication. The user interface program handles all input and output of the simulation program. It must convey the correct input command line selected by the user to the program, and must provide the correct results from the program to the user. To support different variety of simulation tools, we need to establish a smooth and efficient link (*application interface*) between the interface program (PIX) and its supporting simulators and analysis tools.

Beside communication, tool integration is also a function of a user interface program. Through the proper control of the application interfaces, integration of simulations can be achieved using an optimum design environment. For instance, PIX conveys the resist dissolution parameters obtained by the execution of PARMEX to SAMPLE for profile simulation. Results of the simulation can then be plotted and examined on line in separate PIX windows.

The design of the interface of PIX with PARMEX is described in section 3.2. It serves as a prototype for other program interfaces. Techniques for future application interfaces are discussed in section 3.3.

3.2 Interface with PARMEX

PARMEX is a resist dissolution analysis tool for automating the process of determining quantitative models for resists. It can be run in a batch mode or interactively to analyze the experimental development data. The messages and results from PARMEX are displayed through PIX. In addition, the processed data can be plotted in separate PIX windows for examination.

Unlike the interface of SIMPL-DIX with SIMPL-2 [8], the interface between PIX and PARMEX is accomplished through the use of the virtual terminal scheme instead of the Inter-Process Communication (IPC) facilities [9]. The tradeoff of the virtual terminal method is explained in detail after the general structure section.

File Descriptors

UNIX is a file-oriented operating system. Everything in UNIX looks like a file: terminals, disks, tape drives, specialized devices look and act like files. Because of this, the UNIX kernel must abstract a large amount of data for the user. UNIX does this by giving the user a *file descriptor* that the user uses for each transaction about a given file (or device).

Pipe

A pipe is a pair of file descriptors (one reads, one writes) that implements a FIFO (*first-in, first-out*) buffer. This means that a pipe allows users to write data into one file descriptor and read them from the other.

General Structure

Figure 3.1 shows the flowchart of the connection of PIX application interface. With each invocation of PIX, a pair of connected file descriptors (*pipe*), *FD[0]* and *FD[1]*, is created by calling the routine *GetFD()* which is explained in the next section.

After the creation of the pipe, the process running PIX is split into two processes using the *fork()* command. The parent process continues to run PIX, and is pending for the establishment of the link to the simulation program (PARMEX). On the other hand, the child process is changed to a non-graphic mode, and its standard input and output file descriptors are redirected to one of the file descriptors, *FD[0]*. After the execution of PARMEX, using *execl()*, the parent process, PIX, can communicate with PARMEX through *FD[1]*. PARMEX then reads from *stdin* and writes to *stdout*. This approach sets up a two-way stream communication channel between the two processes.

Figure 3.2 shows the application interface schematics. After the user has selected a command and its operands, PIX makes up the command line and writes the command line to the file descriptor *FD[1]*, which will be received by the child process, PARMEX, through *FD[0]*. Similarly, after PIX has sent a command line to PARMEX, it waits for the execution of that command. As soon as the execution is complete, PARMEX sends its results to the standard output, *FD[0]*, which will be received by PIX through the other end of the file descriptor, *FD[1]*.

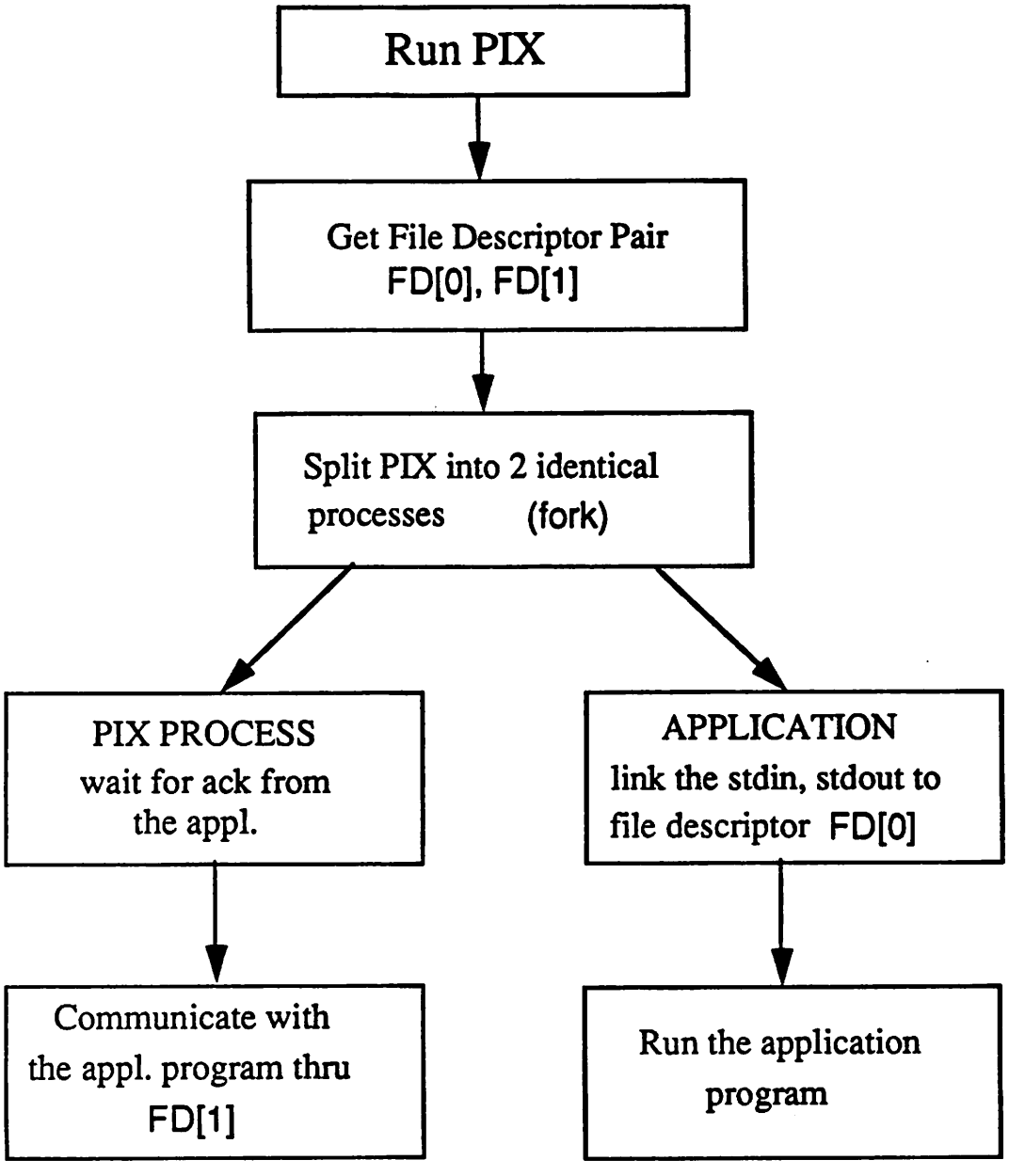

Figure 3.1 Flowchart of the Connection of Application Interface

The output action routine, *PIXWrite*, the input action routine, *PIXRead()*, the error handler routine, *PIXErrorHandler()*, and the connection termination routine, *PIXQuit()* are all in the file "PIX_interface.c". These routines work in the same way as those in the SIMPL-DIX program.

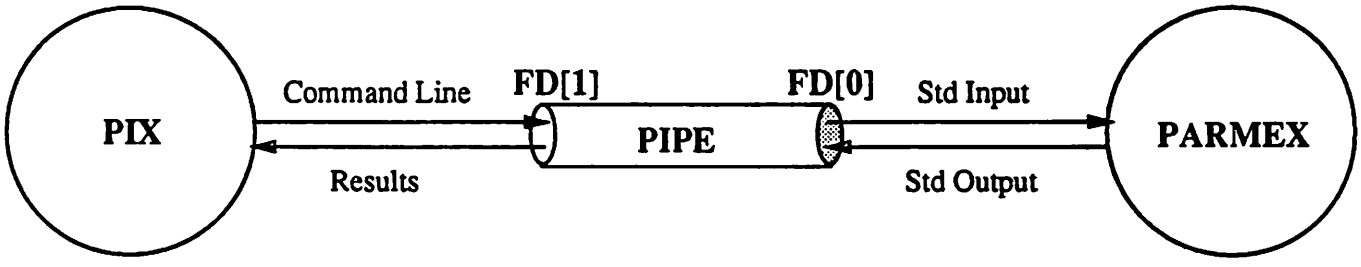

Figure 3.2 Application Interface Schematic

Virtual Terminal Methodology

There is a major disadvantage using IPC with socket. It requires the simulation program using standard I/O operations such as *printf()* and *scanf()* in C language. That is, a new-line character has to be sent, or the file buffers have to be flushed. Flushing file buffers can be accomplished using *fflush()* system call in C (*flush()* in FORTRAN) in the program. If the standard I/O operations are not used, a *fflush()* statement has to be added for every output statement in the simulation program. A user interface programmer cannot change his or her client, the simulation program. Moreover, this change creates a lot of overhead into the application program.

To solve the above problem, the virtual terminal method which works similar to the socket method was adopted. Figure 3.3 shows the *GetFD()* procedure to get a pair of file descriptors. *GetFD()* checks for the availability of a pair of virtual I/O device terminals (pty & tty windows) in the same machine system. If there is a pair available, *GetFD()* reserves them and copys their terminal file descriptors into variables *FD[0]* and *FD[1]* respectively.

The terminal file descriptors are used in the same way as that of IPC socket to generate the link between PIX and its supporting simulators. They carry the same concept as UNIX tries to make everything look like a file. The pair of file descriptors are connected together such that data written into one file descriptor can be accessed from the other file descriptor, and vice versa. Since PARMEX sees a virtual I/O device terminal instead of a socket, it automatically flushes the file buffer every time there is a write statement. As a result, no *flush* statement is required. In addition, the virtual terminal method can bypass the terminal-type check in some of the simulation programs.

The virtual terminal method may cause an infinite loop if running in a heavily used machine. The *GetFD()* routine is running in a loop until a pair of vacant, virtual user terminals is found in the local machine. Every computer system has a limit of tty windows which can be supported. If lots of windows and devices have been opened or used in the background, the routine will not be able to find a pair of vacant, virtual terminals and will loop forever. As a result, PIX is also pending forever waiting for the acknowledgement of the connection. This problem can be solved by setting a timer. If the connection cannot be established by a pre-defined duration of time, PIX will give up the connection and send out an error message to the user.

3.3 Future Interface with Other Simulation Programs

The application interface with PARMEX can be used as a prototype example for future interface with other simulation programs. With the virtual terminal scheme, a reliable communication channel can be established easily.

In general, there are two requirements for a new application program in order to be linked to PIX. First, the executable file of the program has to reside on the same machine as that of PIX. The global variable *Application* in the file "TEST_callback1.c" specifies the location of the executable file of the application program on the system. Both the virtual terminal and IPC socket schemes can only establish a communication link locally. To link to application programs residing on multiple machines, the Remote Procedure Call (RPC) package [10] needs to be used instead.

Secondly, a program prompt which consists of the name of the program followed by a '>' character is required. "*parmex* >", the prompt for PARMEX, is output when PARMEX is done with the previous command execution and is ready for the next

PARMEX is done with the previous command execution and is ready for the next command.

The '>' character in the prompt acts as a clue for PIX. The *PIXRead()* routine waits and reads all the data sent by PARMEX through the pipe after a command has been sent for execution. The routine continues reading until a '>' character is read. The whole prompt, "*parmex*>", is also displayed on the PIX window following all the PARMEX results and messages to show users that PIX is ready for the next selection.

Since '>' is used as a special clue for PIX, new application programs cannot have any messages or results that contain this special character. Otherwise, the prompt mechanism will not work properly.

```
/*****************************************************************
* GetFD:
*       Virtual Terminal method to get a pair of file descriptors.
*       Return FD[0] & FD[1].
*****************************************************************/

GetFD(FD)

int FD[];

{
        int i;
        int  FOUND = 0;
        char *p;
        int fd;
        char ptystr[11];
        char ttystr[11];
        strcpy(ttystr, "/dev/tty00");
        strcpy(ptystr, "/dev/pty00");

        for (i='p'; i<='z'; i++)
        {
          for (p = "0123456789abcdef"; *p; p++)
          {
                ptystr[8] = i;
                ptystr[9] = *p;
                if ((fd = open(ptystr, O_RDWR, 0)) > 0)
                {
                   ttystr[8] = i;
                   ttystr[9] = *p;
                        if ((FD[0] = open(ttystr, O_RDWR, 0)) > 0)
                        {
                          FD[1] = fd;
                          FOUND = 1;
                        }
                }
                if (FOUND)  break;
           }
           if (FOUND)  break;
        }
}
```

Figure 3.3 Virtual Terminal Procedure

Chapter 4

Resist Parameter Extraction using PIX-PARMEX

4.1 PARMEX

PARMEX is a resist dissolution analysis tool for automating the process of determining quantitative models for resists by correlating resist dissolution data with exposure-state information. PARMEX combines dissolution measurement data such as that from the Perkin-Elmer Development Rate Monitor (DRM) with SAMPLE simulation of the resist exposure-state to determine plots and models of dissolution-rate versus exposure-state. The flow diagram for this data analysis process on PARMEX is shown in Figure 4.1. PARMEX commands can be grouped according to their functions. Some of the major groups are signal processing, transformations and model fitting. A library of resist dissolution models and fitting routines are used to determine the resist model parameters.

A typical analysis session would include conditioning data, plotting data to locate bad zones, simulating exposure-state and fitting a dissolution model. The first step in the session is signal conditioning to minimize noise in the raw, experimental data. Plotting shows the nature of noise in the interferometric signals, and various spike and smooth operations can be interactively applied to define an analysis procedure for all zones. Then, the smoothed intensity versus time plots for each zone corresponding to different exposure doses are generated and examined. The plots help the user to identify bad data zones. The bad zones are then deleted from the current working stack in the program. Transformations from intensity to depth to rate are carried out. SAMPLE is then used to simulate the resist exposure-state (*CALC PAC*). Data for

exposure-state versus depth and rate versus exposure-state are obtained. The data for rate versus exposure-state is plotted, and different models are selected for fitting. The best fitting model and its parameters can then be used for profile simulation.

4.2 PIX-PARMEX Plotting Features

Graphical data are plotted in separate PIX windows. The plot window can be resized, printed and iconized for later display. PIX can automatically open multiple plot windows and allow users to examine them on the screen simultaneously. As a result, comparison and analysis among different sets of data can be done easily.

One example of showing how the plotting capability in PIX helps PARMEX users in analyzing the data while running the program occurs in signal conditioning. Figure 4.2a and 4.2b shows the Shipley XP-8843 data for intensity versus time from a DRM before and after signal conditioning. The intensity versus time signal can have significant noise and so conditioning is essential. First, recognizable spikes are cleared. Then, the signal is smoothed by averaging with neighbors to remove additional signal roughness. However, the determination of the threshold for spikes and the filter size is an iterative process. Therefore, being able to select condition parameters and examine plots before and after the execution of "SPIKE" and "SMOOTH" commands on line can greatly facilitate the process.

Another example of using interactive graphic advantageously is in fitting models to rate versus exposure-state data. Figure 4.2c and 4.2d show the KTI-895i data fitted with the *mack* and *dill* models [11] respectively. The chosen model is fit to the dissolution rate versus exposure-state data using Marquardt's method of least square error parameter estimation [12]. The coefficient of determination for *mack* to KTI-895i data

is 0.87 while that of *dill* is 0.98 (1.0 : perfect fit). By comparing the two plots, we can determine the *dill* model is a better dissolution model for the data. With the concurrent plotting capability, users can easily compare the models and then determine which model best fits the experimental data.

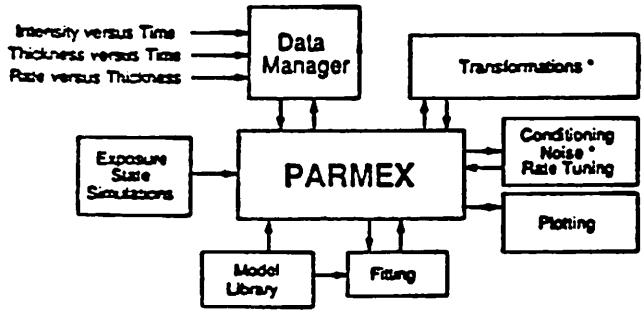

Figure 4.1 Flow diagram of PARMEX.

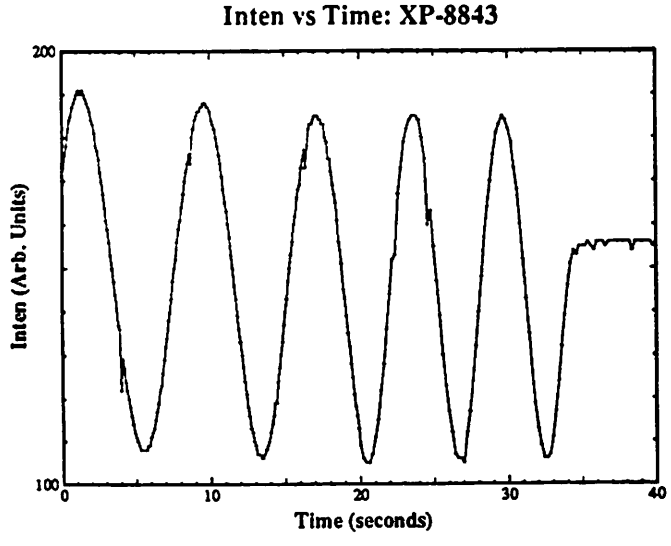

Figure 4.2a Intensity versus Time plots of KTI-895i before signal conditioning.

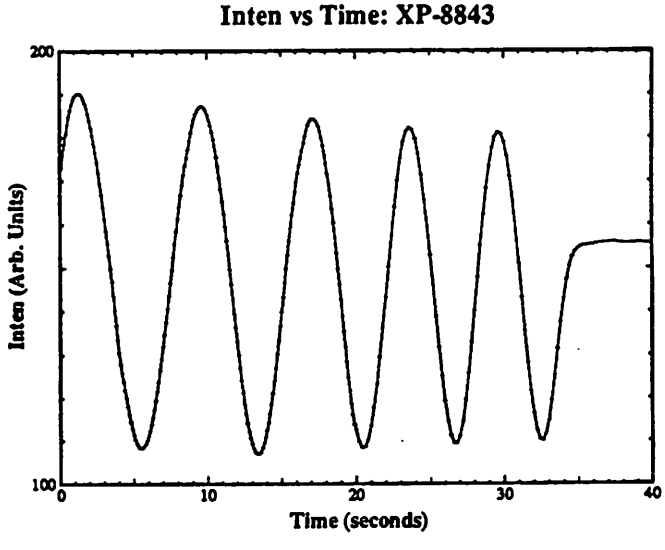

Figure 4.2b Intensity versus Time plots of KTI-895i after signal conditioning.

4.3 Modeling of KTI-895i, an i-line Resist

Resist Development Model

KTI-895i is a photoresist designed for i-line imaging. The resist is composed of a diazo-type photoactive compound and a novolac resin [13]. The dissolution rate of this resist is based upon a dissolution inhibition mechanism. For this reason, we chose the modeling approach of Dill and Kim [14] which has historically worked well for this type of material.

Dill model:

$$\text{Rate} = e^{(E_1 + E_2 m + E_3 m^2)} \qquad (1)$$

Kim model:

$$\text{Rate} = \frac{1}{R_1^{-1}(1 - m \cdot e^{-R_3(1-m)}) + R_2^{-1} \cdot m \cdot e^{-R_3(1-m)}} \qquad (2)$$

where m is the normalized remaining photo-active compound (PAC) concentration,

E_1, E_2, E_3, R_1, R_2, R_3 are the fitting parameters.

This resist also exhibits a surface inhibition effect which can be described by the *Kim* surface inhibition model:

$$\text{Rate} = \text{bulk–rate}\ (1 - (1 - (R_5 - (R_5 - R_6)\, m))) \cdot e^{-\frac{Z}{R_4}} \qquad (3)$$

where bulk-rate as described by either equation (1) or (2) is the dissolution rate near the bulk, the region not affected by surface and substrate inhibition,

R_4, R_5, R_6 are the fitting parameters.

KTI Processing and Results

Resist samples of thickness about 1.1 μm were prepared by spin coating the resist on a wafer at a speed of 4500 rpm for 30 seconds. The spin was followed by a pre-exposure bake of 90°C for 60 seconds to remove excess solvent. The exposures were done with an i-line stepper at a wavelength of 365 nm. There were three wafers in the experiment consisting of 45 exposure zones with doses ranging from 40 mJ/cm^2 to 300 mJ/cm^2. Following the exposure, the resist was baked on a hot plate at a temperature of 110°C for 60 seconds. The resist was developed at 21°C in KTI 945 Immersion developer.

PIX-PARMEX allows the user to process zones and wafers separately or collectively through the use of stacks. The data from each wafer was processed in a similar manner to the session mentioned in section 2 up to the exposure state calculation with *CALC PAC*. Since the noise level varied from wafer to wafer, the data from each wafer were then processed individually. After the data from the first wafer were pushed into the current stack, they were processed, stored and then popped out from the stack. The same technique of pushing and popping data zones was applied to the second and third wafers. After processing the third wafer, the data from good zones on all three wafers were combined. Zone 16 of wafer 1 and zones 14 to 17 of wafer 2 were very noisy, especially near the surface of the resist, and were removed from the stack. The wafer ID together with the zone number is used as the identification for a zone in the stack during the execution of *add* and *delete* zone commands. A maximum of 70 zones in the current stack is permitted.

CALC PAC was executed to run SAMPLE to simulate the resist exposure-state for all of the combined data in the stack. The 45 data zones were then correlated with

the exposure-state information from SAMPLE. The correlated data in the format of dissolution rate versus normalized remaining photo-active compound (PAC) concentration is shown in Figure 4.3a. Data zones with high exposure dose are located at the top left side of the curve. As the curve goes down to the right, the exposure dose of the zones decreases. The data zones with low exposure doses are more noisy than the high-dose zones. In addition, the dissolution rates close to the surface of the resist are much slower than that of the bulk for both high-dose and low-dose zones due to surface inhibition.

The fitting process consisted of two phases. First, the bulk data for all wafers was fit with both the *dill* model and the *kim* model. The *subrange* command in PIX-PARMEX allows the user to select the data depth range to be processed. In this case, to ensure that only bulk data was processed, a range of 0.4 to 0.8 μm from the surface of the resist was selected. We found that the *dill* model provided a better fit to the data and was used to obtain values for E_1, E_2 and E_3 of equation (1). Figure 4.2d shows the fit to the bulk data using the *dill* model with parameters $E_1 = 7.1$, $E_2 = 1.2$, $E_3 = -10.2$, and a determination coefficient of 0.98.

In the second phase, Kim's function in equation (3) was combined with the *dill* model to account for the surface inhibition. This new model formed by the combination of the Kim and Dill functions has been termed the *3E3R* model. Figure 4.3b shows the fitted model plotted at the surface (Z=0) and the bulk of the wafer. The model with $E_1 = 7.1$, $E_2 = 1.2$, $E_3 = -10.2$, $R_4 = 1280$ A, $R_5 = 0.46$, $R_6 = 0.07$ fits the experimental data well for the whole depth range as demonstrated by a coefficient of determination of 0.95.

SEM pictures with different exposure times ranging from 250 msec to 375 msec for a 0.5 micron equal lines and spaces mask pattern from a GCA 2145 stepper (0.45 NA) were obtained from KTI. A resist thickness of 1.23 microns was used on silicon substrate followed by a pre-exposure bake of 90°C for 60 seconds. Then, the exposed resist was developed in KTI 945 Immersion developer for 180 seconds. Figures 4.3c and 4.3d show the SEM views of the resist after development, with exposure times of 325 msec in which the linewidth is exactly 0.5 micron and 275 msec, a 15.4% under-exposure.

The dissolution model (*3E3R*) and resist parameters were put into SAMPLE for profile simulation. The exposure of the resist was modeled using ABC parameters obtained from KTI of A = 0.52 μm^{-1}, B = 0.081 μm^{-1}, C = 0.015 cm^2/mJ, and a refractive index of 1.81. A resist thickness of 1.26 microns on a silicon substrate with the same processing conditions as the experiment was used in the simulation. Figures 4.3e and 4.3f show the simulation profiles using exposure doses of 65 mJ/cm^2 and 55 mJ/cm^2.

It was found that the simulation using a dose of 65 mJ/cm^2 resulted in the line being exactly on size which matched with Figure 4.3c (325 msec). The dose is low compared to the exposure in msec and the normal working dose for several reasons. We had assumed in the analysis a good coupling case (odd multiples of $\lambda/4$), and we discovered later that the patterning experiment used the weak coupling case (even multiples of $\lambda/4$). In addition, the A, B, C's were measured on one system, DRM exposure on a second system currently out of calibration, and the pattern exposure on a third system. The problems with dosimetry can be alleviated by normalizing to a nominal dose required to produce a feature exactly on size. Figure 4.3g shows that the

exposure latitude curve from the simulation fits quite well with that from the experimental results.

4.4 Modeling of XP-8843, an Acid Hardened Resist

Resist Chemistry and Development Model

Shipley XP-8843 resist is composed of a poly(p-vinyl) phenol resin, a melamine crosslinking agent, and a photo-acid generator [15]. Upon exposure, the acid generator produces hydrobromic acid. During the subsequent post-exposure bake, the acid catalyzes a crosslinking reaction between the melamine and the resin. The extent of the crosslinking determines the dissolution rate during development.

A kinetic model was derived relating the acid generated during exposure to the extent of crosslinking during the bake [16]. Then, the dissolution rate could be related to the number of crosslinking events during the bake by:

$$\text{Rate} = R_o \left(1 - \frac{CE}{C_o}\right)^{\alpha}$$

where CE is the number of crosslinking events, and

R_o , C_o and α are fitting parameters.

The number of crosslinking events is given by:

$$CE = \sum_{k=2}^{n_s} (k-1)\ C(n_s, k)\ C_{as}^{k}\ (1 - C_{as})^{n_s - k}$$

where n_s is the number of crosslinking sites (six for melamine) and

C_{as} is the concentration of activated crosslinking sites.

AHR Processing and Results

Spin coating at a speed of 4000 rpm for 30 seconds followed by a pre-exposure bake of 100°C for 1 minute was used to prepare the resist samples. The exposures were done at a wavelength of 248 nm. Following the exposure, the resist was baked for 60 seconds to drive the crosslinking reaction. One wafer was baked at temperature 130°C while the second wafer was baked at 140°C. The resist was developed in 0.135N MF312 developer at 21°C.

After processing the raw reflectivity versus time data, PIX-PARMEX was used to transform the data into the format of dissolution rate versus resist thickness. Figure 4.4a shows the XP-8843 data at various stages during PIX-PARMEX execution. An AHR customized version of SAMPLE was then executed to simulate the acid generation during exposure and its diffusion and reaction during exposure bake. PIX-PARMEX allows various dissolution-rate versus exposure-state correlation to be made. The experimental data was correlated with both the acid concentration as well as the crosslinking site concentration.

In Figure 4.4b and 4.4c, the dissolution-rate versus the normalized acid concentration plot is compared with the dissolution rate versus the concentration of activated crosslinking sites. Data from the two wafers with different baking temperatures produces two separate curves when correlated with acid concentration. The dissolution rate is, however, a single valued function with respect to the concentration of activated crosslinking sites. This confirms that the extent of crosslinking, as measured by C_{as}, and not the acid concentration is fundamental to the resist dissolution. The dissolution rate versus crosslinking site concentration was fitted against the above AHR model named *rfncs* in PARMEX. The model with R_o = 361.4 A/sec, C_o = 7.9, and α = 9.3

compares well with experimental data for bake temperatures of both 130°C and 140°C as demonstrated by a coefficient of determination of 0.985.

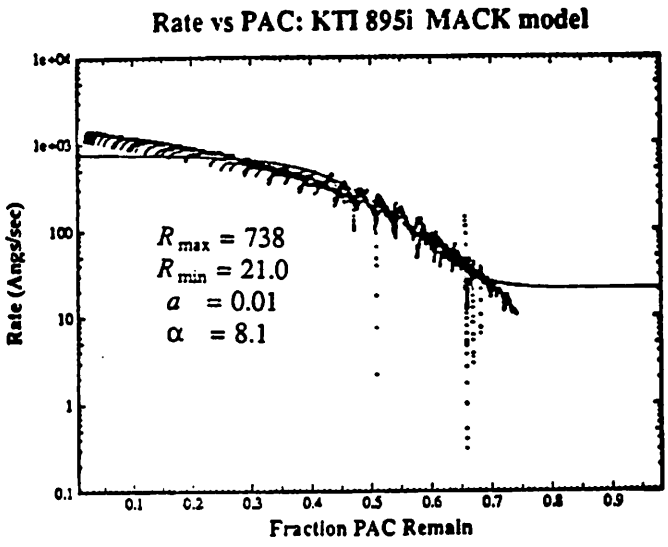

Figure 4.2c Rate versus PAC & *mack* model (Determination Coeff. = 0.87)

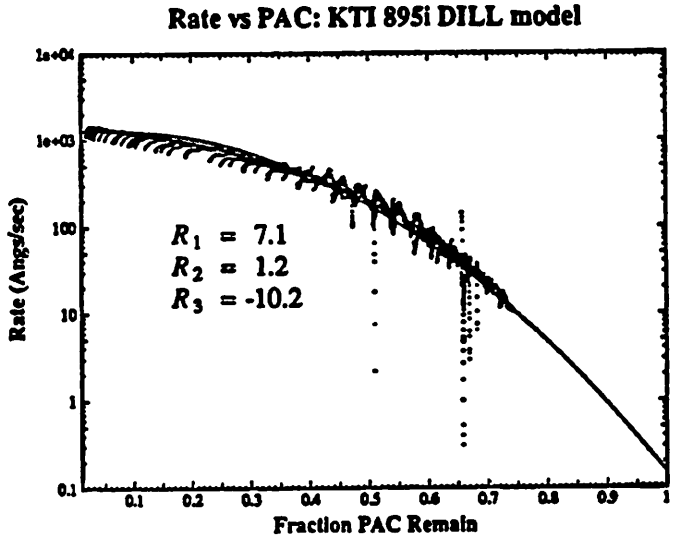

Figure 4.2d Rate versus PAC & *dill* model. (Determination Coeff. = 0.98)

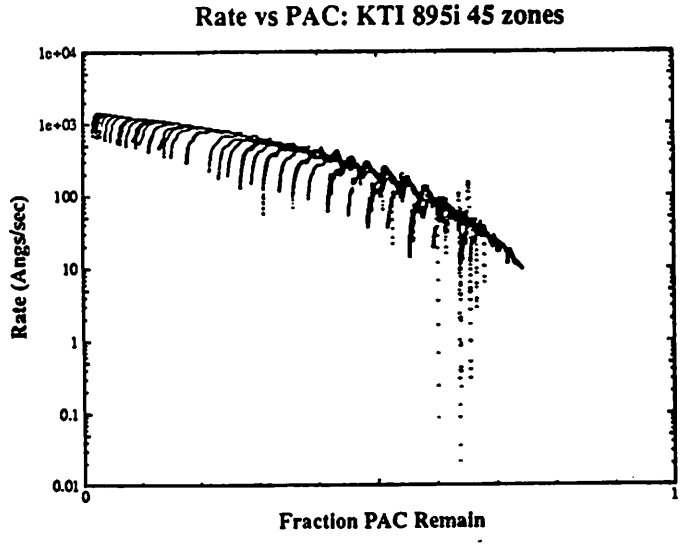

Figure 4.3a Rate versus PAC for KTI-895i.

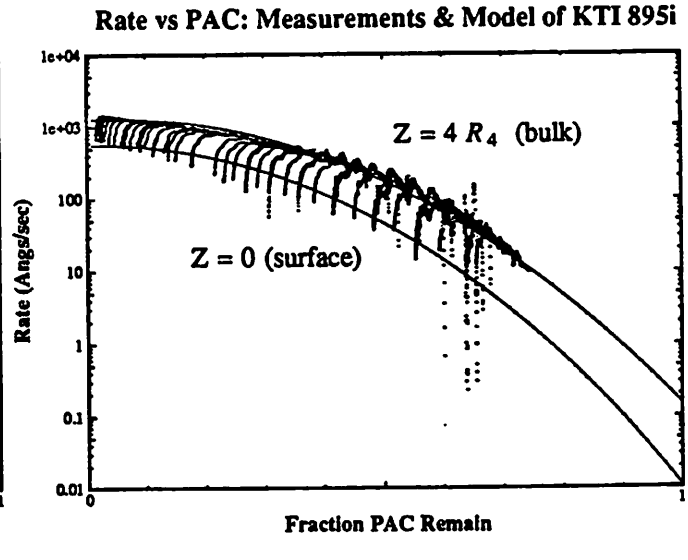

Figure 4.3b Dissolution Rate Measurements and Model of KTI-895i for the whole data range.

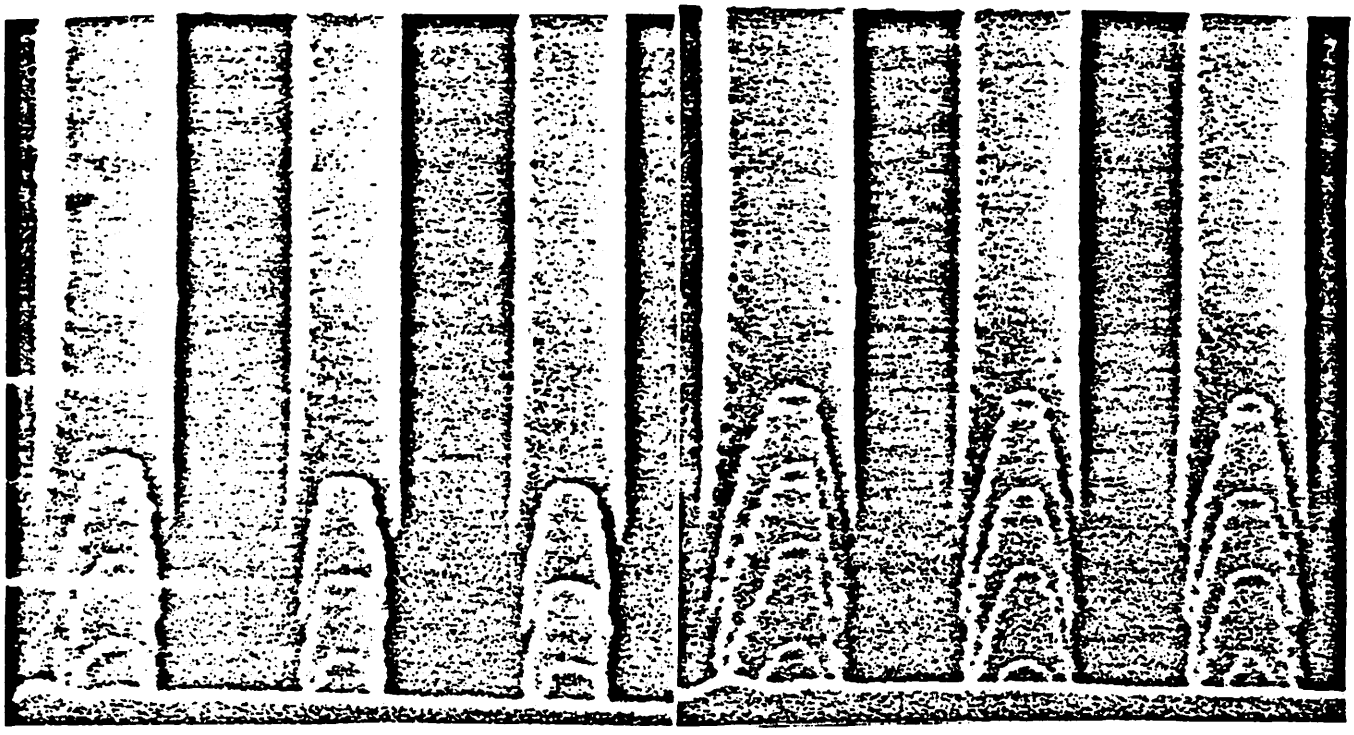

Figure 4.3c SEM picture of KTI-895i 325 msec, 180 sec development time.

Figure 4.3d SEM picture of KTI-895i 275 msec, 180 sec development time.

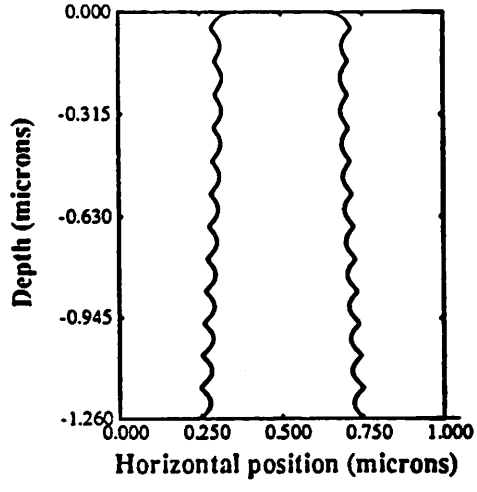

Figure 4.3e Resist Simulation Profile 65 mJ/cm^2, 180 sec development time.

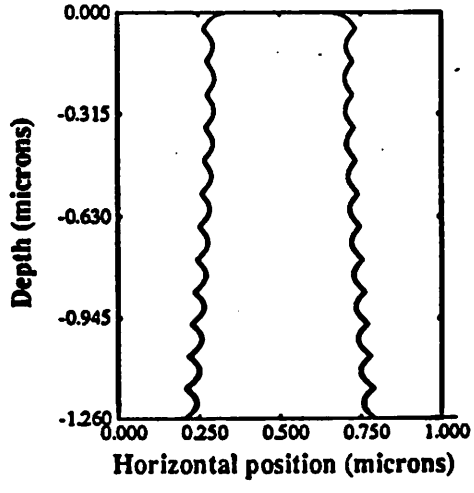

Figure 4.3f Resist Simulation Profile 55 mJ/cm^2, 180 sec development time.

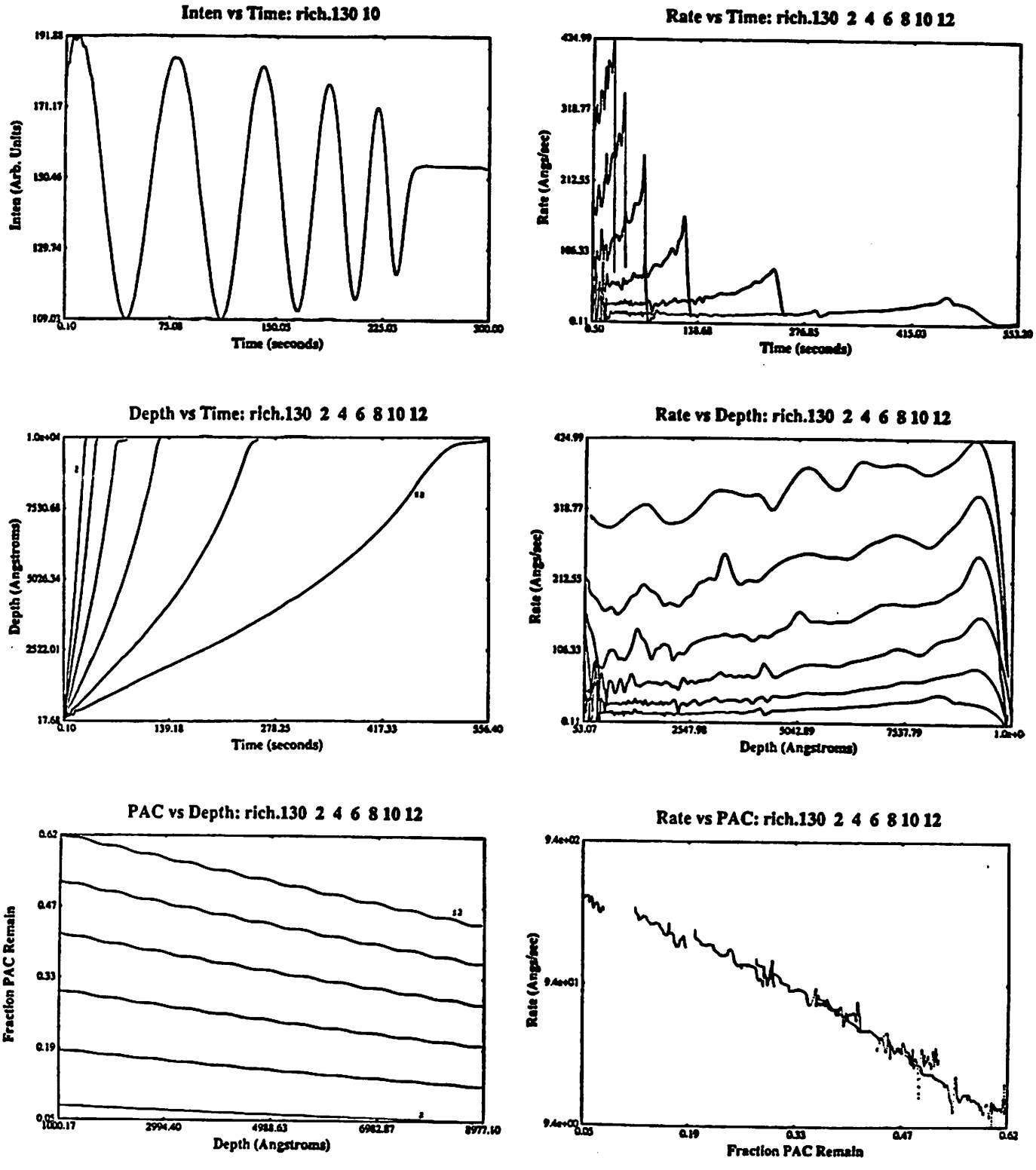

Figure 4.4a Shipley XP-8843 data at various stages.

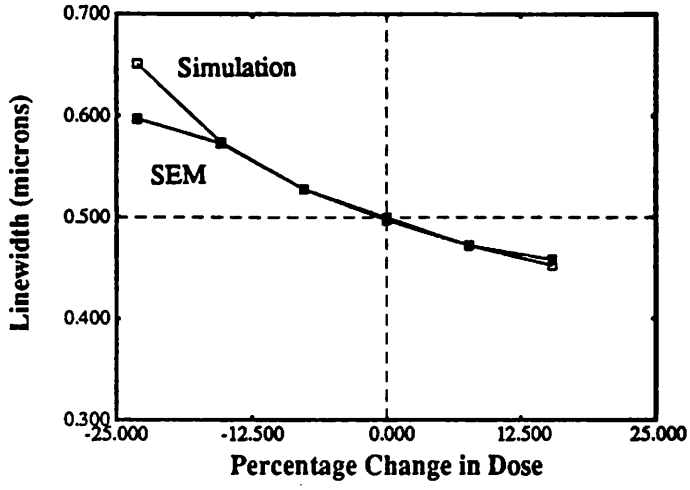

Figure 4.3g Exposure Latitude Curves for KTI-895i.

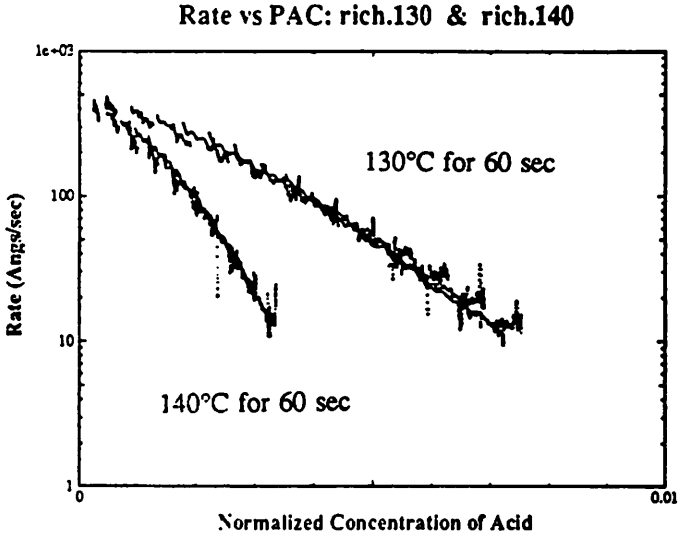

Figure 4.4b Dissolution Rate versus Normalized Acid Concentration.

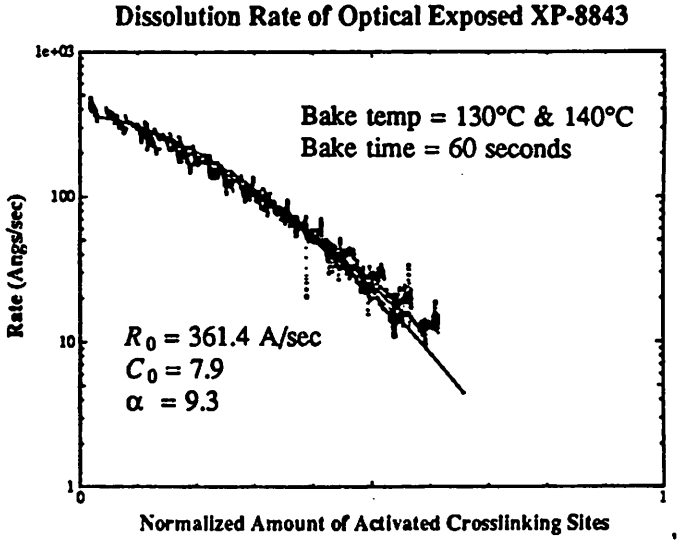

Figure 4.4c Rate Measurements & Model for XP-8843. (Determination Coeff. = 0.985)

Chapter 5

Future Extensions

A graphical user interface (PIX) has been developed for the resist parameter extraction program, PARMEX in X windows on workstations. Expansions and improvements are needed to make the software a better user interface tool. The extension of PIX to support the process simulation program, SAMPLE and the replacement of Athena widgets to MOTIF widgets are two possible future work for PIX.

No major change to PIX is required to link it to the SAMPLE program. The changes needed to set up the application interface between PIX and SAMPLE are described in Chapter 3.3. The links to SAMPLE related plotting programs such as Pdraw and Contour can be set up in the same way as that of Drawplot.

SAMPLE is a fairly large program with 25,000 lines of codes. Unlike PARMEX, it has many more commands, and most of them require multiple operands. The simple, dialog-selection objects described in Chapter 2.4 will not be able to serve the needs. As a result, new dialog boxes in the format of complex tables have to be created for PIX-SAMPLE. There are no table-like widgets in the Athena Widget Set. Creating tables using Athena widgets can be done. However, the format of the table is restricted to one-column entries.

The recent SAMPLE-3D program [17] contains the etching and development machines. It has fewer commands and requires fewer operands than that of SAMPLE Ver 1.7a. Without creating new dialog objects, a prototype version of PIX-SAMPLE3D can be developed within a short period of time.

The Athena Widget set contains simple, primitive widgets and provides limited functionalities. The table widget mentioned in the last paragraph is an example. On the other hand, MOTIF consists of many widgets that have higher performance and more functionalities. In addition, MOTIF has gained industry acceptance and is becoming the main GUI standard. Major computer firms such as IBM, HP, DEC and MENTOR have selected MOTIF widgets to develop their software products. As a result, most users are familiar with the "look and feel" of MOTIF widgets. The modification to MOTIF widgets will help PIX gain user acceptance and popularity.

REFERENCES

[1] W. R. Bell, P. D. Flanner III, C. Zee, N. Tam, and A. R. Neureuther, "Determination of Quantitative Resist Models form Experiment," *SPIE Advances in Resist Technology and Processing V,* Vol. 920, pp. 382-389, 1988.

[2] S. N. Nandgaonkar, G. Addiego, T. E. Berger, J. L. Reynolds, and A. R. Neureuther, "SAMPLE User Guide," *Electronics Research Laboratory, Department of EECS, Univ. of Calif.,* Berkeley, CA 94720, Version 1.6a, February 1985.

[3] J. McCormack, P. Asente and R. Swick, X Toolkit Intrinsics and X Toolkit Athena Widgets - C Language Interface, X Window System Version 11, Release 3. Digital Equipment Corporation, Western Software Lab. 1988.

[4] Adrian Nye, The Definitive Guides to the X Window System Vol I - Xlib Programming Manual for Version 11, O'Reilly & Associates, Inc., 1989.

[5] OSF/MOTIF Programmer's Guide. Prentice Hall, Englewood Cliffs, 1990.

[6] Hsi-Cheng Wu, "SIMulated Profiles from the Layout - Design Interface in X *SIMPL-DIX*" *Memorandum No. UCB/ERL M88/13,* January 1988.

[7] Kenny K. Toh, "Algorithms for Three-Dimensional Simulation of Photoresist Development " *Memorandum No. UCB/ERL M90/123,* p. 278-281, December 1990.

[8] K. Lee, *SIMPL-2* (SIMulated Profile from the Layout - version 2), Electronics Research Laboratory, U.C. Berkeley, July 1985.

[9] Phil Lapsley, Kurt Pires, "C in the UNIX Environment," Electronics Research Laboratory, U.C. Berkeley, p. 72-78, 25 September, 1984.

[10] Alexander S. Wong, "An Integrated Graphical Environment for Operating IC Process Simulators" *Memorandum No. UCB/ERL M90/67,* 25 May, 1989.

[11] W. R. Bell, P. P. Flanner, W. L. Chu, T. Doi, A. Chiu, "PARMEX User's Guide," *Electronics Research Laboratory, Department of EECS, Univ. of Calif.,* Berkeley, CA 94720, Version 1.0, p. 1-5, December 1989.

[12] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *J.Soc. Indust. Appl. Math.,* vol. 11, no. 2, p. 431, June 1963.

[13] R. Barber, T. Brown, B. Katz and J. Greeneich, "A Production Worthy 0.5 Micron Process for IC Fabrication," *KTI Chemicals Corporation Publication, 1990.*

[14] D. J. Kim, "Characterization and Modeling of Positive Photoresist," Memorandum No. UCB/ERL M84/65, 17 August, 1984.

[15] R. A. Ferguson, J. Hutchinson, C. A. Spence, and A. R. Neureuther, "Modeling and Simulation of a Deep-UV Acid Hardening Resist," *J. Vac Sci Technol. B,* to appear Nov/Dec 1990.

[16] R. A. Ferguson, C. A. Spence, Y. Shacham-Diamond, and A. R. Neureuther, "Modeling and Simulation of Multiple Chemical States in Photoresist Materials," *SPIE Advances in Resist Technology and Processing VI, vol. 1086,* pp.262-273, 1989.

[17] E. W. Scheckler, K.K.H. Toh, D. M. Hoffstetter, and A. R. Neureuther, "3D Lithography, Etching, and Deposition Simulation (SAMPLE-3D)," *To be presented in 1991 Symposium VLSI Technology, Oiso, Japan.*