1994

# Equipment control and process control in semiconductor CIM systems

Steve K. P Wong
*San Jose State University*

## Recommended Citation

Wong, Steve K. P, "Equipment control and process control in semiconductor CIM systems" (1994). *Master's Theses*. 883.
DOI: https://doi.org/10.31979/etd.z3w7-gf3u
https://scholarworks.sjsu.edu/etd_theses/883

Order Number 1359066

# Equipment control and process control in semiconductor CIM systems

Wong, Steve Kwok Po, M.S.

San Jose State University, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# EQUIPMENT CONTROL AND PROCESS CONTROL IN SEMICONDUCTOR CIM SYSTEMS

A Thesis
Presented to
The Faculty of the Department of Mechanical Engineering
San Jose State University

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science

by
Steve K.P. Wong
August 1994

APPROVED FOR THE DEPARTMENT OF MECHANICAL ENGINEERING

Dr. Albert Hu

Dr. Imin Kao

Dr. Ji Wang

APPROVED FOR THE UNIVERSITY

Serena N. Stanford

# ABSTRACT

## EQUIPMENT CONTROL AND PROCESS CONTROL IN SEMICONDUCTOR CIM SYSTEMS

by
Steve K.P. Wong

A recent study [1] shows that control system failure accounts for fifty percent equipment failure in semiconductor fabrication. In response to this study, an overview of today's Computer Integrated Manufacturing (CIM) systems for equipment control and process control in fabs is presented here. Three CIM systems developed in different environment are investigated. They are the Strategic Cell Controller developed in SEMATECH, the semiconductor consortium industry, the MIT Computer Aided Fabrication Environment developed in academia, and the Microelectronics Manufacturing Science Technology CIM system developed by Texas Instruments under the industry and government's support. These CIM systems are being studied in the areas of computer architecture, user interface, distribution of software and hardware, supporting resources, development tools, and development and implementation cost.

Six equipment and process control systems and three CIM implemenation projects will be presented here to highlight the adaptability and flexibility of control systems in today's CIM environment.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AMS | Adaptable Manufacturing System of Standford's Cluster Based Fab |
| APC | Advanced Process Control Module of ControlWORKS |
| ASIC | Application Specific Integrated Circuit |
| ATP | Adaptable Tool Platform of IBM |
| AVP | Advanced Vacuum Processors of MMST |
| BCAM | Berkeley Computer Aided Manufacturing System |
| CAFE | Computer Aided Fabrication Environment |
| CEM | Common Equipment Model |
| CIM | Computer Integrated Manufacturing |
| CLOS | Common Lisp Object System |
| CMP | Chemical Mechanical Planarization Process |
| CPU | Central Processing Unit of Computer |
| CTC | Cluster Tool Control Module of ControlWORKS |
| EWMA | Exponential Weighted Moving Average |
| FRC | Fairchild Research Center of National Semiconductor Corporation |
| GCC | Generic Cell Controller of the University of Michigan |
| GEM | Generic Equipment Model |
| GUI | Graphical User Interface |
| MIMO | Multiple Input Multiple Output |
| MISO | Multiple Input Single Output |
| MIT | Massachusetts Institute of Technology |
| MMS | Mass Manufacturing System |

| | |
|---|---|
| MMST | Microelectronics Manufacturing Science and Technology Program of Texas Instruments |
| MVRbR | Multivariate Run by Run Control |
| NSC | National Semiconductor Corporation |
| PFR | Process Flow Representation of CAFE |
| PMC | Process Module Control Module of ControlWORKS |
| POM | Process Operation Monitor |
| RbR | Run by Run Control of MIT |
| RTP | Rapid Thermal Processing of MMST |
| RtR | Run to Run Control of the University of Michigan of U.C. Berkeley |
| RTSPC | Real-Time SPC System |
| SCC | Strategic Cell Controller of SEMATECH |
| SECS | SEMI Equipment Control Standards |
| SFC | Shop Floor Control |
| SPC | Statistical Process Control |
| TI | Texas Instruments |
| TMC | Transport Module Control Module of ControlWORKS |
| UIB | User Interface Builder of MMST |
| VLSI | Very Large Scale Integrated Circuit |
| VM | Virtual Machine |
| WIP | Work in Progress |

# INTRODUCTION

As shown in Figure 1 [1] below, semiconductor manufacturing industry is in a very unique position in contrast to many other manufacturing industries - the processes are highly complex and uncertain. A wafer typically goes through hundreds of processing steps and tens of pieces of equipment before it is completed. It requires complex relationships between process parameters at various steps and from step to step in order to perform process control. Because the processes are often difficult to control, typical process yield is about 40 to 80 percent. Due to difficulty in equipment and process control, process equipment utilization is also low. It is not uncommon to have below 50% overall usage of process equipment.

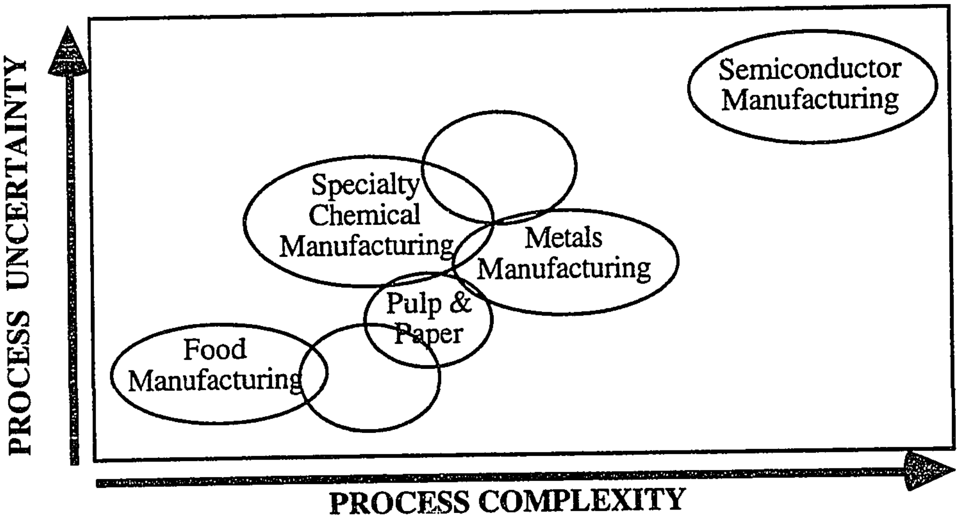

Figure 1: Manufacturing industry Perspective in terms of process uncertainty and process complexity

In the current semiconductor manufacturing industry, different kinds of computer software are used to improve the intelligence and integration of equipment. However, the industry may be too small to support a dedicated supplier base for software development and, thus, must used shared manufacturing software [2]. Because of the usage of different software developed on different software platform, the scope of equipment integration must include software integration as well.

Today's manufacturing systems for the semiconductor industry have evolved over the past 20 years. The current technology status of these systems has shown that the industry suffers from a variety of limitations [3]. First, they are costly to implement and maintain. Every factory is different, and the systems ( even the so-called standard products ) must be extensively customized to support a particular manufacturing site. Second, they are inflexible, often dictating a compromise manufacturing approach rather than supporting a desired one. Third, many systems are difficult to use and interrupt rather than aid the manufacturing users. For this reason, needed information about the manufacturing process is often inaccurate or missing altogether. Fourth, the systems are poorly integrated. They have been developed over many years by many people in an environment of changing requirements and changing technology [3].

Another problem of today's semiconductor manufacturing systems is that the development has been dependent on systems to catch the mistakes of a less qualified work force rather than relying on skills and discipline of their work force. This results in needing a more complex manufacturing system. Moreover, manufacturing system is viewed as a burdensome add-on that has to be separately justified rather than an essential part of the initial capital investment just like process equipment. Furthermore, the rather short length of service and instability of the work force means that the team is seldom responsible for multiple implementations, carrying their learning and effectiveness with them [3].

Equipment is usually designed with little understanding of its relation to process stability and control, and its relation to the manufacturing system. Semiconductor manufacturing equipment has evolved over the past thirty years from relatively simple tools to complex systems. It is highly specialized and expensive. The equipment is usually produced by small companies or small work groups of a company [3]. These companies and work groups design new equipment based on past experience and extrapolation, rather than a common knowledge base and a formal design discipline. Their equipment design is very subjective. Integrating equipment into an existing manufacturing system is another challenge in the industry. Also, in most cases, only empirical understanding is available for processing modeling design. This limits the design of predictable process equipment.

This thesis consists of three chapters. First, an overview of the software architecture and user interface of three semiconductor manufacturing systems is given. In this chapter, the Strategic Cell Controller of SEMATECH, the Microelectronics Manufacturing Science and Technology CIM system of Texas Instruments and the Computer Aided Fabrication Environment of MIT are described. Second, five equipment and process control systems are reviewed. They are the Adaptable Tool Platform of IBM, process and equipment control systems of MMST, ControlWORKS of Texas Instruments, Generic Cell Controller of the University of Michigan, the Run-by-Run process control system of MIT and the BCAM system of U.C. Berkeley. Third, three CIM systems implementation projects are reviewed. They are the Run-by-Run control of the CMP process developed by Fairchild Research Center of the National Semiconductor Corporation and San Jose State University, FABCON from Japan, and the Adaptable Manufacturing System of Stanford's cluster-based fab. The paper is ended with concluding remarks from the author. Please refer to the List of Abbreviations on page vii for the abbreviations used in this thesis.

# CHAPTER 1: THE CURRENT CIM SYSTEMS FOR SEMICONDUCTOR FABRICATION

Traditional CIM systems tend to depend on one huge software program running on one single main frame. Whenever the main frame is down, the information flow of the manufacturing facility is crippled. The gigantic software program is difficult to customize and update due to the complexity and the length of the coding. It can only be modified through a painful debugging process. In some cases, the software program is written by the main frame developer and comes with the main frame. Then, software program is updated when the main frame developer comes up with a new version. In either case, these traditional CIM systems are not made flexible to the manufacturing facility, rather, the facility has to make itself flexible to accustom the CIM system. For example, one of the aerospace manufacturing companies in southern California used a MRP II system running on an IBM AS400. The planning function of the system restricted each operation description to a length of 40 characters, which is not enough for the engineers to put down their instructions. Moreover, the traditional CIM systems tend to have an abstract ASCII terminal for the user interface. It makes the training of the CIM system user to be difficult, and it makes the implementation of the CIM system to be even more agonizing and expensive.

Modern semiconductor CIM systems use distributed computing. Compared with the traditional CIM systems, a distributed system provides a less hierarchical software structure so that both hardware and software applications can be used in a more independent fashion. Multiple users can run the same application simultaneously from any node on the network. Therefore, distributed computing is fast becoming an important trend in CIM systems in various industries. The advantage of distributed computing includes the following:

a. The CIM network will be more reliable and robust to the failure of hardware or software. When failure of one part of the distributed hardware or software occurs, the network can still function because it does not solely rely on one single piece of hardware or software like the conventional CIM systems.

b. The distributed software system makes the customization and debugging easier.

c. Most of the systems consist of graphical user interface. It is much more intuitive than ASCII terminals, and thus, less training is needed for new users.

d. Distributed computing also makes data transparent to all users on the network; therefore, it reduces redundancies of data entry.

e. Distributed computing enables the use of heterogeneous hardware; therefore it better utilizes existing hardware.

The down side of distributed computing is that it is highly dependent on the behavior of the network and the messaging protocol. In the following semiconductor CIM systems to be examined, the general approach is distributed computing.

## 1.1. SEMATECH STRATEGIC CELL CONTROLLER

The SEMATECH[1] SCC (Strategic Cell Controller) is a CIM system that is developed for the SEMATECH member companies. It is supported by the semiconductor industry. The main objective of the SCC program is to supply a common and flexible framework for cell controllers for the members of SEMATECH. This common framework provides an adaptable platform for software vendors to develop application software packages and equipment interface for different modules of SCC. The framework is flexible that SCC can control and monitor manufacturing cells at remote locations. Also, it has virtually unlimited expandability. C++, Motif, Xwindows and CELLworks (a manufacturing cell development software package from FASTech) are the primary development tools of the SCC.

### 1.1.1. SCC Architecture

Figure 2 illustrates the architecture of SCC CIM system. SCC operates between the Shop Floor Control (SFC) system and the equipment. SECS II is the primary equipment message protocol for the SCC. The servers in Figure 2 represent either a device interface or application software. Regardless of the physical location of these servers, they are available to serve at any nodes on the network once they have been integrated into the system. Servers can exchange information with each other using a standard set of messages that is developed for the SCC program. Additional servers can be easily integrated into the SCC architecture as long as the server's developer employs the standard SCC messaging protocol. In addition to the development of the distributed client-server architecture and

[1] SEmiconductor MAnufacturing TECHnology consortium is a non-profit organization representing majority-owned and controlled U.S. semiconductor manufacturing equipment, processing materials, software, and service companies such as Hewlett Packard, IBM, National Semiconductor, Applied Material and Varian.

messaging protocol, standardization of development methodology of cell applications is also the main focus of the SCC Program [4]. Much effort has been made to create a common equipment model for the factories of the SEMATECH members so that the SCC CIM system is able to satisfy the needs of each member. Member companies can still maintain their proprietary technology, and yet be able to use a common framework to adapt a variety of application software and device interface.

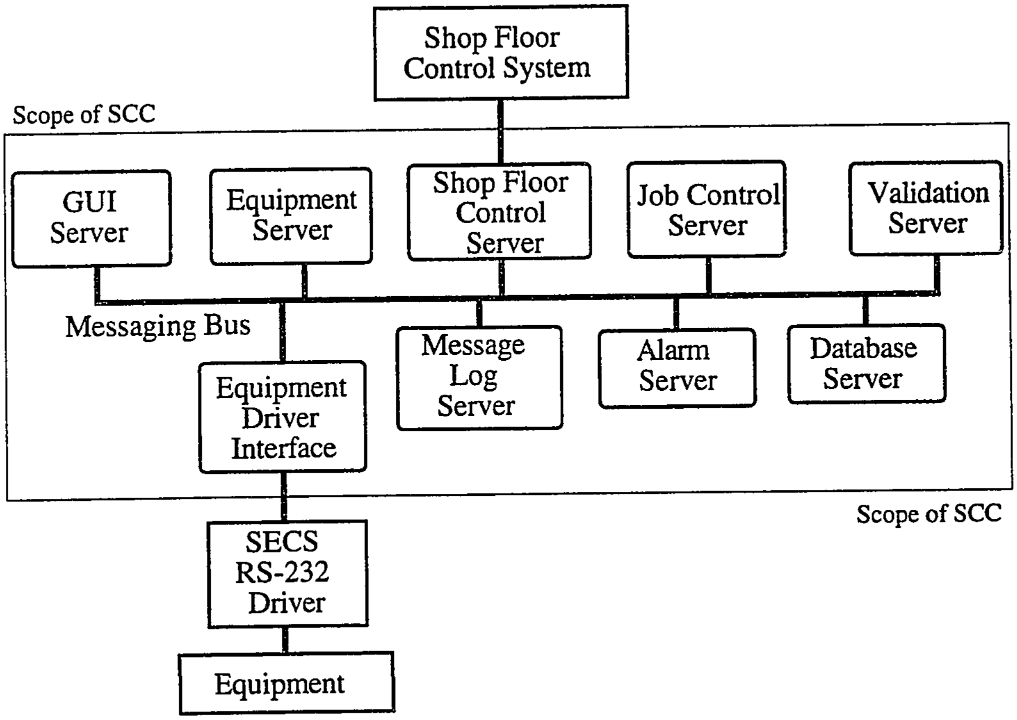

Figure 2: SCC CIM architecture. The servers are chosen as an example. Depending on necessity, other applications, devices and equipment can be integrated into the framework above. SCC operates between equipment and the SFC (Shop Floor Control) system.

Thinking of the SCC Program as a mother board of a computer , the architecture of the program can be described by Figure 3. The different slots represent the plug-in boards of the different servers, or applications, that are within the system. Servers can obtain information from the factory and feed back to the management level, and execute process on the shop floor following the management's request. SCC can have as many applications as the factory needs and view all of them as one single application. The idea of this is similar to a personal computer running different applications under Microsoft Windows. For example, Figure 3 contains 4 slots for different servers. As long as the servers can be inserted into the mother board, the servers would function as part of the SCC. Virtually, this SCC mother board can have an unlimited number of slots. This common framework also makes the technology transfer between member companies easy.

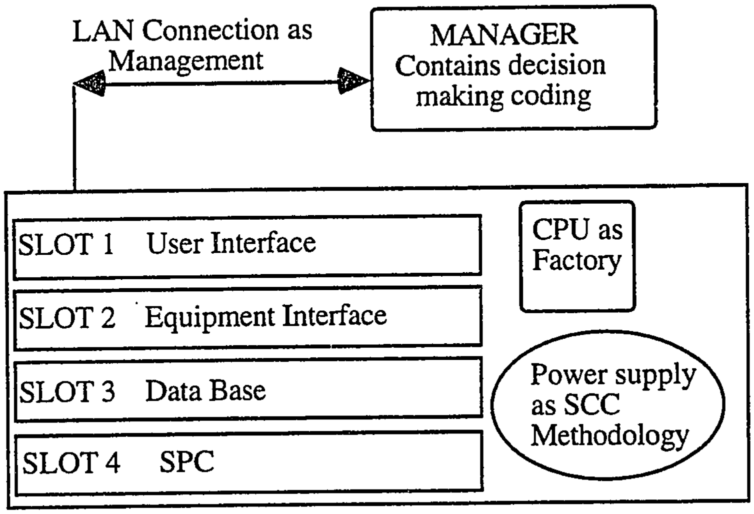
LAN Connection as Management
MANAGER Contains decision making coding
SLOT 1 User Interface
SLOT 2 Equipment Interface
SLOT 3 Data Base
SLOT 4 SPC
CPU as Factory
Power supply as SCC Methodology

Figure. 3: SCC CIM architecture is similar to a PC mother board.

### 1.1.2. SCC Graphical User Interface

Having the same look and feel for the user interface in the fab area is a challenge since each piece of fab equipment tends to have its own style of user interface. They vary from pure ASCII terminals with monochrome screen to graphical user interfaces with thousands of colors. It is not an easy task for fab operators to adopt many different user interface styles.

SCC developers attack this phenomena by standardizing styles of different user interface. This common Graphical User Interface ( GUI ) framework employed by the SCC program is based primarily on the SEMATECH's SCC User Interface Style Guide. This Style Guide provides guidelines and conventions for developing common OSF/Motif-style user interface, which is a very common UNIX based window manager. The Style Guide also provides guidelines for the software architecture of the GUI server. Within SCC, the GUI server makes the appearance of client application user interface consistent and performs in a predictable manner. The logic of the SEMATECH's SCC User Interface Style Guide has been evaluated and approved. Using such a guideline for GUI development accelerates the design and implementation effort [5].

## 1.2. TEXAS INSTRUMENTS MICROELECTRONICS MANUFACTURING SCIENCE & TECHNOLOGY CIM SYSTEM

The Texas Instruments MMST (Microelectronics Manufacturing Science & Technology) Program is supported by both the industry and the government. The goal of the MMST Program is to create a factory that can provide fast (cycle time) and economical (cost) production of a variety of devices (flexibility) with first pass success (quality) [6]. The MMST CIM system helps to implement such a factory through integration of wafer processing equipment and the supporting systems for product and process specification, production planning and scheduling, material handling and tracking, and process control. The MMST CIM system is a distributed object-oriented client-server system [6]. The fundamental object-oriented principles help to solve the three biggest problems in software development today: quality, productivity and maintainability. Smalltalk-80 is the primary development tool for the MMST CIM system. Based on an evaluation of several candidate prototypes, the productivity of using Smalltalk for the object-oriented environment was four times that of traditional development in C. For the initial deployment, the MMST CIM system is running on UNIX. Deployment under other operating systems such as OS/2, DOS and MacOS is supported without change to application code. Applications are shielded from differences between the vendor supplied versions of UNIX by the Smalltalk virtual machine (VM), which translates operating system specific services into Smalltalk-80 generic services.

The process control system of MMST will be discussed in Chapter 2 of this thesis. A technological product, named ControlWORKS, has been developed by the MMST Program. It is an equipment and process control software package that is commercially available for the semiconductor industry. ControlWORKS will also be further discussed in Chapter 2 of this thesis.

### 1.2.1. MMST Architecture

The architecture of a distributed object-oriented system is multi-dimensional. The two most important dimensions are the logical and the physical. To illustrate the idea of the logical architecture, Figure 4 is shown as an example. The object classes are arranged in a logical hierarchy. The top of the hierarchy are the most generic classes of the CIM system.

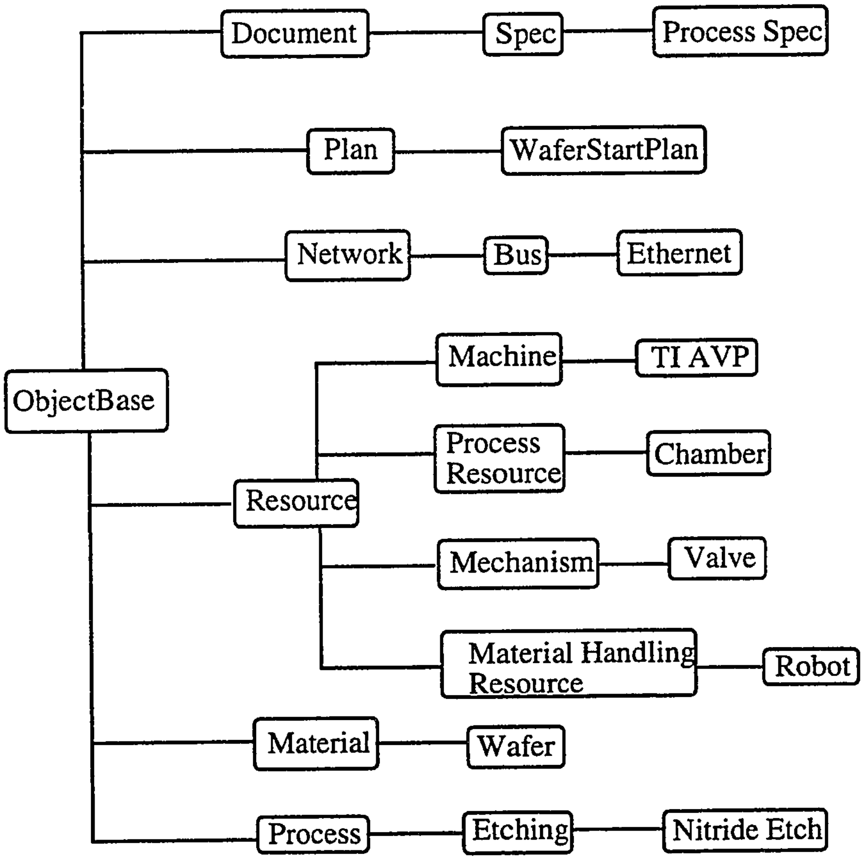

Figure 4: Logical architecture of MMST CIM System

The bottom of the hierarchy contains the most specific classes. For example, the class Resource is generic enough to include a variety of resources such as machine, process resource, mechanism and material handling resource. At the bottom of the Resource hierarchy, the classes are specific enough to point at a particular advanced vacuum processor, chamber, valve and robot arm.

The physical (object instance) architecture is shown in Figure 5. Objects are grouped by the logical architecture and distributed throughout various compute nodes of the fab-wise CIM system. Objects in MMST CIM system are instantiated by the applications at the compute nodes. In other words, tasks are being done by running application software at the compute nodes. At the same time, the information in the object base is updated accordingly. Notice that in Figure 5, both the distributed workstations and the modular process equipment can be viewed as compute nodes. Different classes of objects are instantiated (starting of an object instance) at these compute nodes. For example, the objects in the class of document and plan will be mainly instantiated at the distributed workstations, and the objects in the class of machine and material handling resources will be mainly instantiated at the modular process equipment. Each instantiated object performs certain tasks as services to other objects upon authorized request. Other objects may by involved to accomplish the tasks. For example, the robot must collaborate with both the source and destination objects for material hand-off. Objects interact with each other in terms of parameter exchange. These exchanged parameters are considered as objects as well.

In Figure 5, each modular processing system consists of a variety of processing equipment such as vacuum cassette, vacuum load lock, robot handler, process chamber, vacuum pumping network. All processing equipment is integrated with a host computer (workstation). The host computer within the modular system receives commands from the CIM network and drives each of the processing equipment accordingly. Processes within these compact modular systems are highly automated [6]. These highly integrated modular processing equipment systems are often called cluster tools. In contrast to the traditional wafer processing, where each processing step has to be done at a different location, cluster

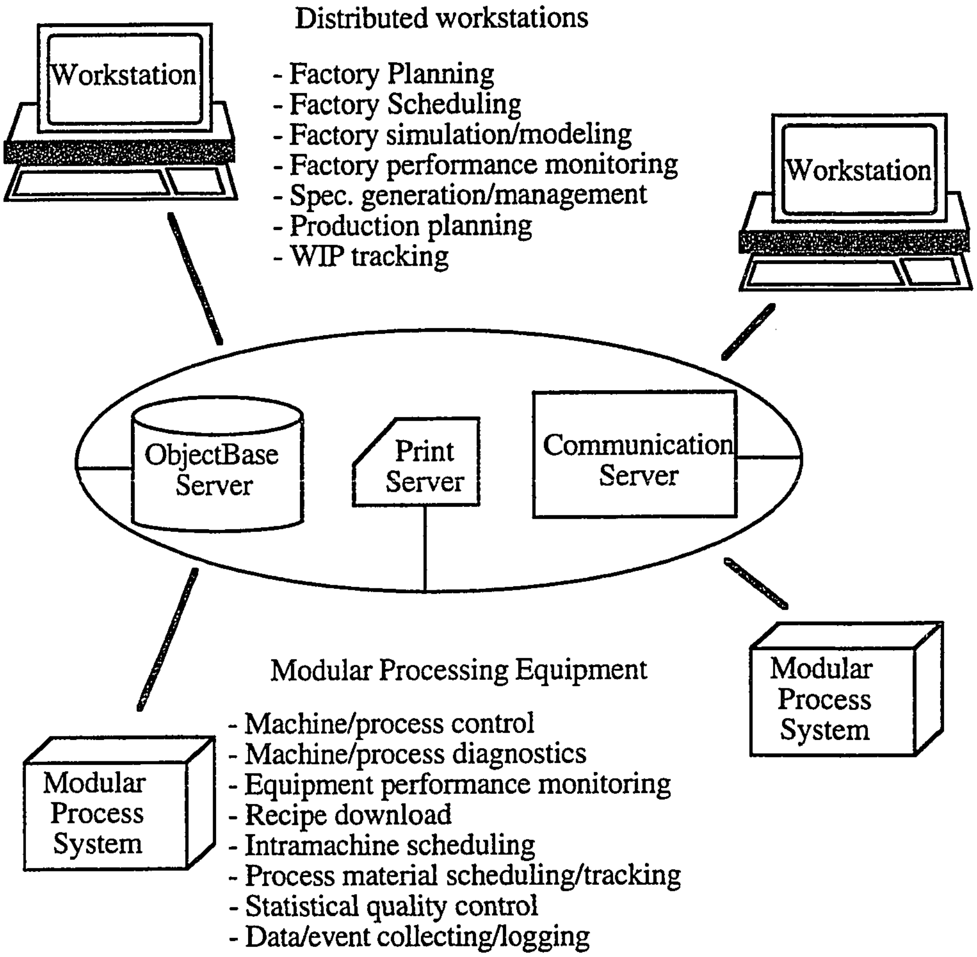
Distributed workstations
Workstation
- Factory Planning
- Factory Scheduling
- Factory simulation/modeling
- Factory performance monitoring
- Spec. generation/management
- Production planning
- WIP tracking
Workstation
ObjectBase Server
Print Server
Communication Server
Modular Processing Equipment
- Machine/process control
- Machine/process diagnostics
- Equipment performance monitoring
- Recipe download
- Intramachine scheduling
- Process material scheduling/tracking
- Statistical quality control
- Data/event collecting/logging
Modular Process System
Modular Process System

Figure 5: Physical architecture of the MMST CIM system. In the shop floor level, the distributed workstations contain applications for factory level tasks. In the manufacturing equipment level, the cubes labeled as modular processing systems represent individual compact wafer processing unit that contain different kinds Advanced Vacuum Processors.

tools can handle multiple wafer processing steps in one location. Maintenance cost is very high for traditional fab where the entire clean room environment is maintained at class 10 or class 1. Using cluster tools, fab maintenance cost can be greatly reduced because only the vacuum wafer cassettes and the vacuum load locks have to be maintained at class 10 or class 1. The fab itself can be kept at a much higher clean room level. The advantages of the MMST fab over the traditional fab are summarized in Table 1 [6].

| **Current Fabs** | **MMST Fab** |
|---|---|
| Fixed, 20,000-wafer | Modular, 100 to 1000 wafer |
| Factory costs > $500M | Factory costs $30M to $ 50M |
| Class 1 Cleanroom | Class 100 Cleanroom |
| 50,000 - 100,000 ft$^2$ | 3,000 - 5,000 ft$^2$ |
| 25 to 50 Wafers Batches | Single-Wafer Processing |
| Batch Sampling | Real-Time Process Control |
| 1-3 months | 5-15 days |
| Mixed Equipment Configuration | Modular Processing Equipment |
| 30% of Processes Use Liquids | Liquid-Free Processing |
| Atmospheric cassettes | Vacuum cassettes |

Table 1: MMST Technology versus Current Technology.

The MMST program has recently been demonstrated on a sub-half micron CMOS flow performed entirely on single-wafer processors. The process successfully used extensive closed-loop real-time process control utilizing a variety of new in-situ process monitors. Actual throughput times of 8.6 days for a 0.35 micron CMOS process have been reported from the MMST fab. In terms of physical process technologies the MMST program was significant in that it used single-wafer rapid thermal processing (RTP) in place of all conventional furnace steps. RTP has high potential to replace most furnace steps over the next decade. Whereas the MMST fab used wet single-wafer cleaning, successful dry single-wafer cleaning has been recently reported using different gas phase approaches at different labs. Also, the use of both cluster tools in general, and specifically open architecture vacuum cluster tools, are rapidly growing in semiconductor manufacturing [7].

### 1.2.2. Graphical User Interface of the MMST CIM System

A User Interface Builder (UIB) has been developed to produce application screens and facilitate the connection of screen components to application objects [6]. The UIB has a What-You-See-Is-What-You-Get (WYSIWYG) style builder and comes with an extensive library of flexible screen components. On the UNIX version of MMST, the X Window System is used for user interface and Motif is used for window manager. The Smalltalk VM supports the use of other X Window managers, as well as OS/2 PM, DOS Windows and Macintosh Windows without change to application source code.

## 1.3. MIT COMPUTER AIDED FABRICATION ENVIRONMENT

MIT CAFE (Computer Aided Fabrication Environment) is a CIM system developed in academia. The design, analysis, and execution of semiconductor fabrication processes are the major functionality of CAFE. Currently, CAFE is being used in managing the Microsystems Technology Laboratory fabs at MIT, Lincoln Laboratory and Case Western University. The primary development tools of CAFE are C and LISP. CAFE will be upgraded by adopting some of the MMST CIM system development approaches such as using Smalltalk.

### 1.3.1. CAFE Architecture

The CAFE architecture, which is shown in Figure 6 [7], consists of three parts: an infrastructure architecture, tool and data integration architecture, and applications. The infrastructure architecture of CAFE defines and provides a set of domain-independent components for use within the system, including the Fabform user interface package and the Gestalt object-oriented database. The tool and data integration architecture defines the conceptual schema and models used to represent and maintain knowledge and information about IC manufacturing within the system, and provides both the user and programmatic interfaces to that information. The applications of CAFE provide support for a wide variety of activities, including not only process but also product, equipment, plant, and personnel management capabilities.

Data integration in CAFE is achieved by a shared process flow representation (PFR). Activity or tool integration is achieved partially by a shared process flow support subsystem which is used by both fabrication and process development support systems. A

PFR is converted from a textual process flows into instance hierarchies of process operations. The process operations are defined in the Gestalt database. The database provides shared, object-oriented access to PFR information for use by both design and manufacturing activities, and can be accessed and updated by all of the tools and applications in CAFE. The process flow editor is not one of the CAFE applications at this stage. Currently, the vi editor of UNIX is used for generation and modification of process flows.

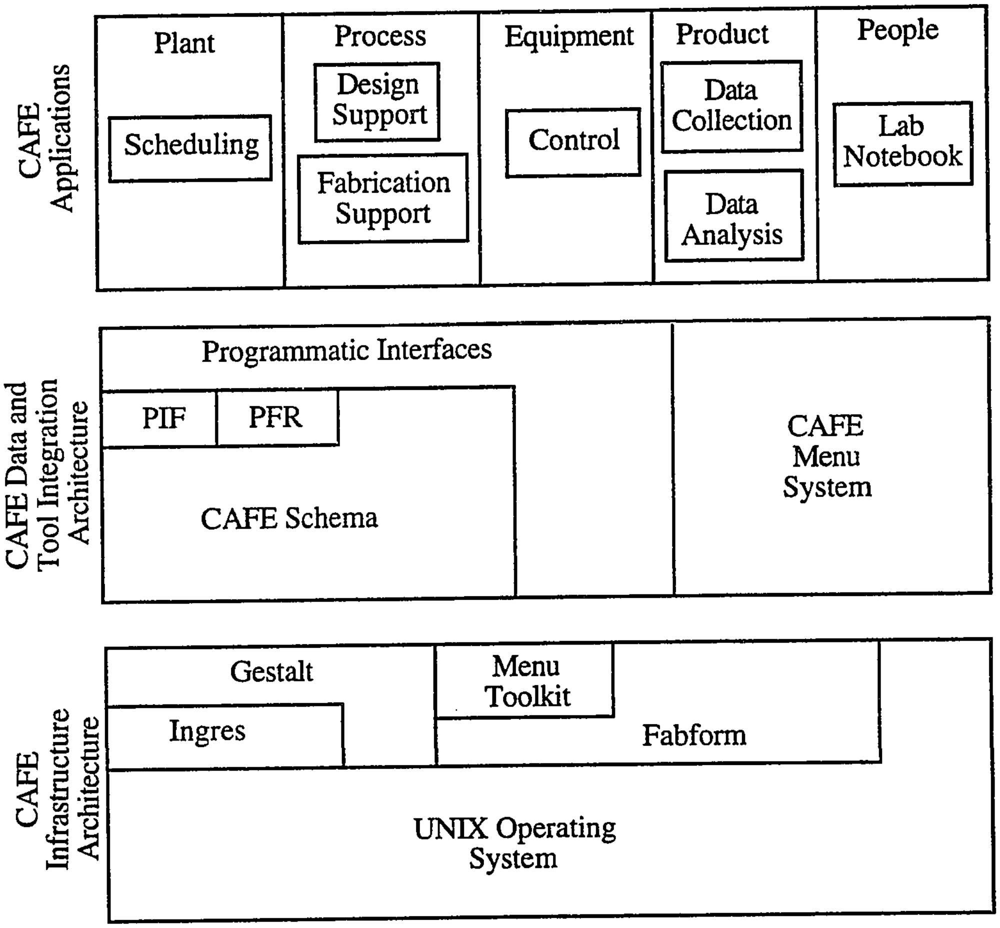

Figure 6: The Architecture of MIT Computer Aided Fabrication Environment (CAFE).

Wafer lots in CAFE are attached to new or existing process flows. Reports such as work-in-progress (WIP) tracking are generated from process flow and lot information. Operations on single or multiple lots can be scheduled for present or future execution by the real-time scheduling system. Using the results from measurements, the PFR models of these processing operations are updated by the fabrication system. Therefore, process development is better informed by information generated during manufacturing.

Process design tools in CAFE provide help during process design, which takes place by incrementally generating a PFR for the process. Design may be done on various levels, including wafer-state-change, physical treatment, or machine settings. Simulation manager supports incremental simulation of the process during development. Process advisor aids in treatment level synthesis, providing analytic model-based estimates for the initial choice and modification of process parameters to achieve process goals. Flow library of basic processing operations for available equipment is maintained in the data base with the necessary physical information such as change of wafer state and machine settings. The data base can be accessed either by matching a defined name or by specification, which are attributes of an operation. The PFR can then be used to support flexible process design styles that are capable of a variety of possible manufacturing implementations. Design rule checks are available to test the adherence of processes to fabrication facility guidelines and policies before execution by the manufacturing subsystem [7].

### 1.3.2. The User Interface of CAFE

The user interface of CAFE is mostly driven by Fabform, a generalized form editor. Menu screens in CAFE can be considered as forms or templates containing a number of blank fields. These forms or templates are Fabforms. Within Fabform, users can have blank fields. These forms or templates are Fabforms. Within Fabform, users can have access to the menu including data entry, predefined choice selection, delete, save, move from one field to another and so forth.

Fabform is designed to run as a separate process and communicates with the associated application via the template file and the parameter file. The template file describes the layout of the menu screen, fields and the types of data that can be entered in the various fields. The parameter file specifies the contents of the various fields defined in the template file. Fabform takes a template and/or parameter file as input and produces a parameter file as an output. Both input and output parameter files may be read from/written to pipes if desired. Once the application program invokes Fabform, it can no longer communicate with Fabform. All interaction between the two processes is done only through the files described above, and the input template and parameter files are read only when Fabform is invoked. The output parameter file is first written as soon as Fabform finishes reading the input parameter file and is updated whenever the user executes a save command, so the application can write the latest changes to the output file.

By incorporating Fabform into the application itself, it becomes possible to associate a function, which may be written in C or Lisp, with each field. The function will be called whenever the value of that particular field changes (the changes can be a complete entry or any individual key stroke) [8].

# CHAPTER 2: EQUIPMENT CONTROL AND PROCESS CONTROL IN SEMICONDUCTOR FABRICATION

Time is important in the semiconductor fabrication industry. A new piece of technology, which required enormous amount of research and development cost and effort, can only take the lead for a short period of time. Therefore, having a flexible and adaptable CIM system is one of the most important factors for success in this industry. Three equipment and process control systems with emphasis on flexibility and adaptability are presented in the following chapters. Among the three systems, the TI ControlWORKS and the IBM Adaptable Tool Platform are two of the off-the-shelf software packages that are commercially available for the industry. The advantage of using off-the-shelf software product is the saving in terms of development time and effort. Besides the ControlWORKS and the Adaptable Tool Platform, the Generic Cell Controller of University of Michigan, RbR Process Control System of MIT, and BCAM of U.C. Berkeley, are also discussed in the following chapters.

## 2.1. ADAPTABLE TOOL PLATFORM OF IBM

The Adaptable Tool Platform (ATP) is developed by the Manufacturing Technology Center (MTC) of IBM. The ATP is written in C, and it runs on OS/2. Similar to the SCC program, the philosophy behind the ATP is provide a common framework for both the users and the software developers. For the users, they can " plug in" as much application software as they need without worrying the compatibility to their existing system. For the software developers, as long as the industry enjoys this common software platform, the demand of application software for this platform will grow.

The ATP system has a modular architecture that allows software components to be independent. It provides supports such as data collection and management, tool configuration, event and error handling, user management and security, network support, message handling, and programming for creating modules. Currently, both the MTC of IBM and other software developers have already created many application modules such as Particle Detector Module, NetBIOS Network Module, Digital and Analog Sensor Module and TCP/IP Network Module for ATP. The functionality of ATP will grow with the number of application modules available. The ATP system also provides a Application Programming Interface (API) for users to develop their own modules.

The architecture of ATP consists of three levels: the control node, tool node and view node. The control nodes are responsible for running the control algorithms. They communicate with the manufacturing equipment through the tool nodes and achieve equipment and process control. The tool nodes are the equipment interface of the ATP system. Information from the control nodes will be collected at the tool nodes and translated into action. The view nodes allow users to monitor the tool nodes and displays real-time data through a configurable graphical user interface. The view nodes also allow users to control the tool node, and thus control any equipment.

## 2.2. PROCESS CONTROL AND EQUIPMENT CONTROL SYSTEM OF MMST

### 2.2.1. PROCESS CONTROL SYSTEM

Figure 7 [9] illustrates the run-to-run control aspect of the MMST process control system. The supervisory controller obtains desired target values and the current states of the predictive model (from process control system) and generates machine settings (from process control system). The regulatory controller will use the machine settings to control the machine in real-time. The observables are collected through the sensor in the machine and sent to the analysis and model adjustment component, which monitors the observed results and makes adjustments to the process model accordingly. The adjusted process model is used to generate machine settings for the next run.

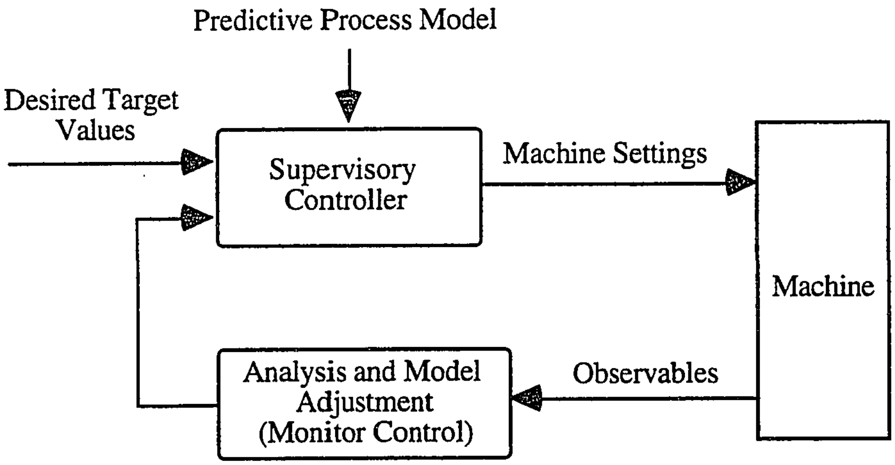

Figure 7: Model-based process control with feedback.

The process control system of MMST acts as a server to the MMST CIM system. It appears to the CIM system as an encapsulated system. It provides services to the CIM system, a client application, through a defined message protocol. This is a typical object-oriented design approach where the client does not need to know how the server operates. Whenever the client (CIM system) requires a process control service from the server (process control system), it sends an inquiry to the server to obtain the corresponding service [9].

A process control model object consists of three attributes: an index that identifies the process in terms of the external environment; a process model that defines the process behavior; and a control strategy that defines how the process model is used to control the process [9]. A control strategy consists of components such as data transform, data analysis test, and optimization strategy. The optimization strategy defines which optimization procedure and objective function are to be used.

The process control model object is interpreted by the process control system. There are seven services supported by the process control system [9]:

Selecting process control models: Process model is selected from an existing process model library based on a given description. The description includes the process to be performed, the process resource, the process material, and indicators of the position of the process in the processing sequence and product flow.

Calculating machine settings: Given a process control model and the target values, the machine settings are calculated based on the current process model (original process model + adjustment), and the constraints and optimization procedure defined by the control strategy.

Creating and storing information: The information of each run is created and stored under an object called run. Run has its own data structure. The Generic Equipment Model (GEM) is responsible for creating run objects that represents process steps.

Measuring and analyzing from run information: Upon completion of the run, post-run analysis is performed based on the measurement data. The control strategy components measurement calculators, data transforms, data analysis tests and tuners are executed in the post-run analysis. Each component produces results when it executes. Analysis is completed when execution of the strategy components is finished.

Measuring and analyzing from sensor information: The procedure is similar to the previous service. Sensor information requires a separate service because, while the data may be available at runtime for a process resource with *in situ* measurements, some processes may require off-line metrology before the analysis can be performed.

Creating and editing process control models: A user interface is provided for engineer to define process control models. The user interface allows the engineer to select a process model or a control strategy for editing.

Displaying process results: A graphical display called Process Operation Monitor (POM) displays a parameter value over time.

### 2.2.2. EQUIPMENT CONTROL SYSTEM

The MMST equipment control architecture consists of four layers as shown in Figure 8 [10]. The Common Equipment Model layer defines the interface to the factory and provides a structure for further specialization. This layer generalizes the machine requirement and interface definition. The Cluster Tool layer allows for implementation on distributed CPUs (VME bus-chassis). At the Plasma layer, further refinement of abstractions adds behavior and plasma specific processing mechanisms such as RF generator, vacuum system and gas systems. The Vendor-Product layer implements specific machine mechanisms for the mechanical architecture of a machine operation, such as gate, inner-chamber and AVP robot arm. This layer includes the GUI control panel for each of the specific machine types developed.

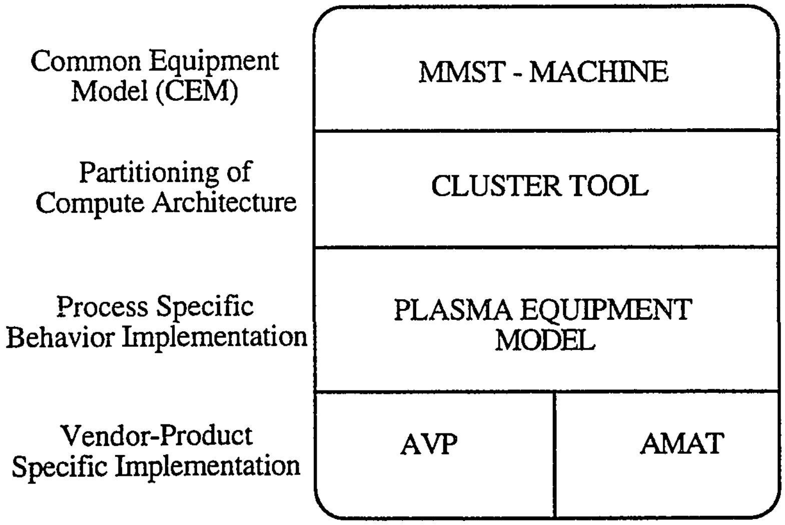
Common Equipment Model (CEM)
MMST - MACHINE
Partitioning of Compute Architecture
CLUSTER TOOL
Process Specific Behavior Implementation
PLASMA EQUIPMENT MODEL
Vendor-Product Specific Implementation
AVP
AMAT

Figure 8: MMST machine control architecture.

CEM consists of four messaging protocols: CIM System Interface, Machine Scheduling, Material Transfer and Process Execution. The CIM System Interface protocol allows the factory to obtain information and control operations. The Machine Scheduling protocol allows the factory to process material on a machine. The Material Transfer protocol allows moving material and executing transfers. The Process Execution protocol allows material to be processed. The development of Machine Performance and Machine Maintenance protocols are still in progress [10].

The CIM System Interface protocol consists of four sections: Process Control Interface, History Interface, Factory Scheduling Interface and Factory/Machine Transfer Interface. The Process Control Interface is a set of messages that interface with the process control system to obtain modified machine settings and report run-to-run data of a machine. The History Interface is a set of messages that allows wafer-to-wafer processing data to be stored. The Factory Scheduling Interface is a set of messages that allows the factory to interact with the machine for determining the processing capability and for committing the machine to specific processing. The Factory/Machine Transfer Interface is a set of messages that allows the factory to interact with a machine to cause material movement between machines [10].

Currently, the MMST Equipment Control system is integrated with thirteen pre-existing machines, nineteen new Advanced Vacuum Processors (AVP) and an Applied Material 5200 PVD cluster tool [10].

## 2.3. CONTROLWORKS OF TEXAS INSTRUMENTS

ControlWORKS is a technological product of the Texas Instruments' MMST program. It is equipment and process control software that combines many of the process and equipment control innovations in the MMST program. Recently, Texas Instruments has made ControlWORKS a commercially available software program for the semiconductor manufacturing industry.

ControlWORKS consists of four components for process and equipment control. They are the Cluster Tool Control (CTC), the Transport Module Control (TMC), the Process Module Control (PMC) and the Advanced Process Control (APC). The CTC provides a user interface, material router, data store and factory interface for a variety of cluster tools. The user interface of CTC can be customized to user's needs. The material router of the CTC includes a scheduling framework that can be configured to fit the existing scheduling algorithms. The TMC is responsible for controlling the central material handlers, vacuum system of the transport module of a cluster tool, cassette elevators and load locks. Also, the TMC provides interface to common hardware components such as robot arms, stepper motors, isolation valves, vacuum pumps, and pressure and temperature sensors. The PMC provides controls to energy sources such as heaters, coolers, RF, microwave, DC power sources, magnetrons, and IR lamps. PMC also supports gas system control that works with a wide variety of plumbing configurations. The APC employs number of different control strategies such as flexible adaptation of models, feedback control, and univariate and multivariate Statistical Process Control to generate equipment settings that lead to desirable process target value.

ControlWORKS was developed with two goals - CIM and Equipment. For the CIM side, the goal is to develop a comprehensive, seamless information and control management software that spans the factory, process and equipment domains. For the Equipment side, there are five emphases: 1) develop a machine control software architecture based on cluster tool topology that can be applied to a variety of hardware control architecture including non-cluster equipment; 2) provide a common representation and development environment from I/O board to user interface screen; 3) integrate with the CIM system to provide capabilities for advanced factory level control; 4) integrate with process control to provide run-to-run process control on a single wafer basis; 5) provide an interface to third party equipment with and without SECS II interfaces.

There are four layers of interfaces to enable the reusability of ControlWORKS. The Common Equipment Model (CEM) layer of ControlWORKS, in compliance with other emerging industry standards such as the Generic Equipment Model (GEM) of SEMI, encompasses the generic concepts of equipment. The Cluster Tool layer allows for physical partitions, communications and database support. The Vacuum layer provides the necessary controller for specific processing concepts such as RF, gas control, vacuum and etc. The Product-Supplier level implements vendor specifics [11].

## 2.4. GENERIC CELL CONTROLLER OF THE UNIVERSITY OF MICHIGAN

The Generic Cell Controller (GCC) is developed in academia. The goal of GCC is to achieve Run-to-Run (RtR) control by coordinating the information flow between process modules [12]. Figure 9 shows the architecture and the basic functionality of GCC. GCC is an expert system that uses the sequential control algorithm stored within a relational database to control a cell. The design guidelines is intended to enable the GCC to be

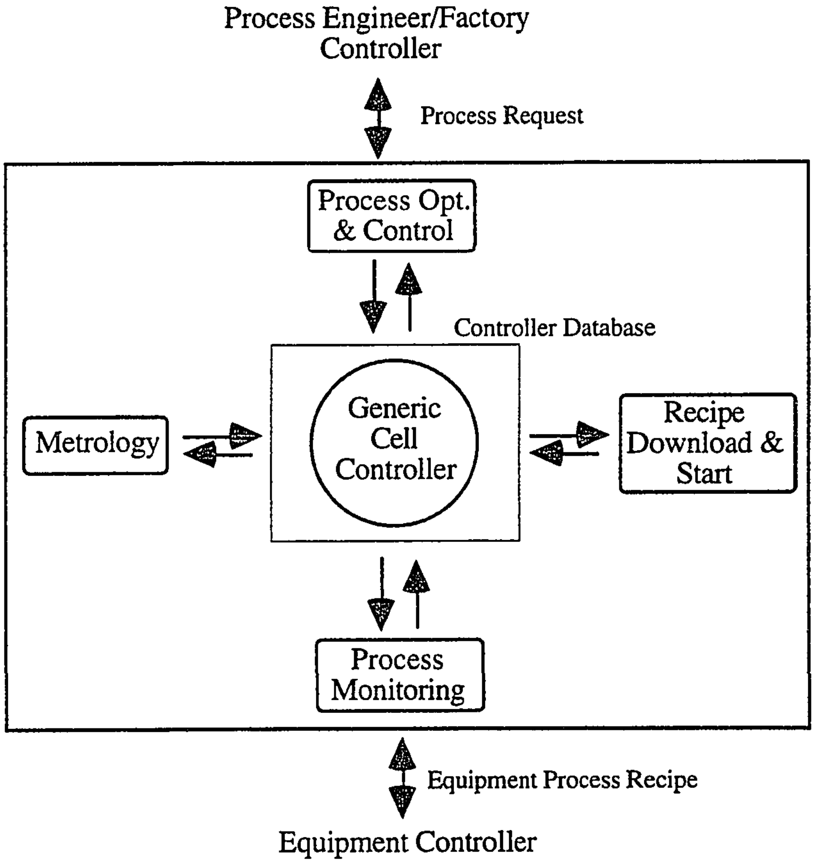

Figure 9: GCC CIM architecture

hardware, software, and communications protocol independent, and is generic to the multitude of semiconductor processes. Currently, GCC is being developed on NEXT computers. Object C is being used in the coding. Object C is a much more object-oriented programming language than C++. It can be viewed as a combination of Smalltalk and C++. The NEXT platform is capable of generating DOS executables. NEXT is also Sun compatible.

There are basically six objects in the GCC framework: modules, message (un)parser, I/O gateway, conductor, timer and database as shown in Figure 10. The term "object" in here is the object in the object-oriented design framework. They communicate with each

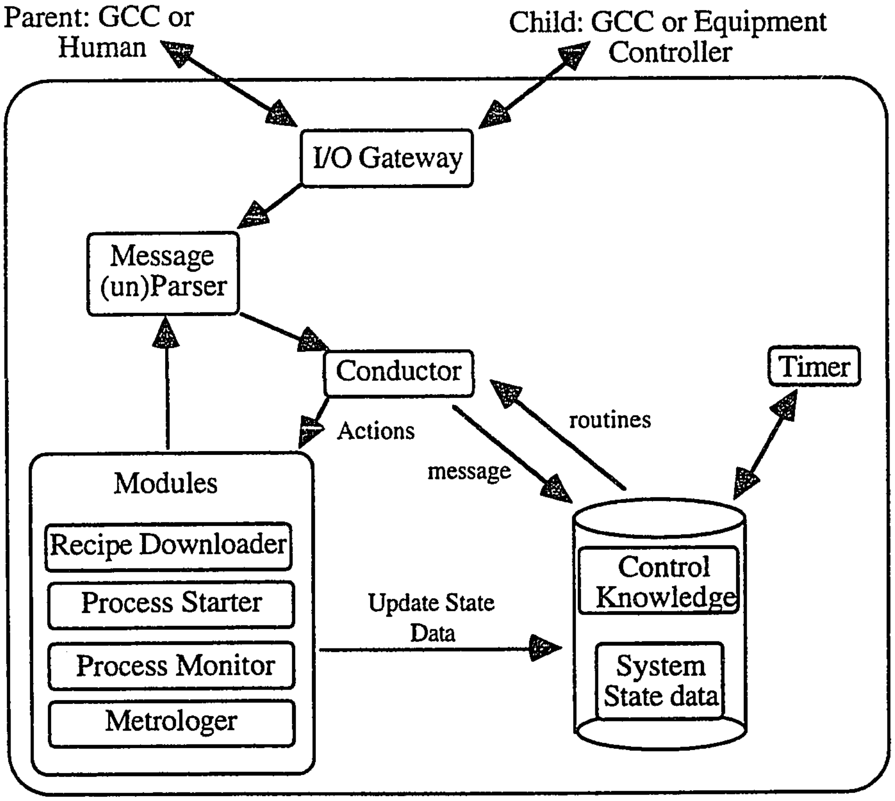

Figure 10: GCC Data Flow Diagram.

other using a messaging format, which is a standard procedure in object-oriented programming. Message is sent simply by pointing an arrow to another object; for example, cout<<"output " in c++, would send the word "output" to an object called cout. Note that the modules, which are objects in the GCC framework, do not communicate with other modules directly. Modules are in passive positions. They would not be executed unless the database tells them to do so. First, the database would accept a request from the user. Then the database would search for an appropriate response, which includes the execution sequence of the individual modules and the corresponding data or setting that goes with the module. Then the conductor takes the inquiries from the database and executes individual modules in the sequence provided by the database. Each module would acknowledge the conductor once the modules have finished their tasks. Then the conductor would execute the next module and so on.

Communication between individual modules and the equipment is established through the message parser and the I/O gateway. The modules sends out a c++ message to the message parser. Then the message is forwarded to the I/O gateway which handles all the TCP/IP networking communication to the “outside” world. However, the individual modules can be written in such a way that they communicate with equipment directly without going through the message parser and the I/O gateway. This could apply in real-time control.

The database has two sides; one is the knowledge base and system state data (in individual data file format). The knowledge base stores the information about the actions on a given message. When a message comes in, the database searches for the corresponding action items. The action items are then translated into specific names of routines, and the corresponding parameters that associate with the routine. Then, this message that consists of the names of the routine and the corresponding arguments in order will be sent to the conductor so that the ordered routines can be executed in sequence. This is called an expert system, which consists of a knowledge base, an inference engine and working memory. The knowledge base defines the rules on how to manipulate and analyze a given set of data that is in the working memory. The manipulation is done through the inference engine. In GCC, the knowledge base is represented by the relational database system, the working memory is the data that is generated by the processes, and the inference engine is represented by the different modules.

In order to put additional modules into GCC, one must follow a set of specifications. This set of specifications is nothing more than the definition of an object. To be more precise, the specification is the class definition in object-oriented programming (similar to the variable and/or type declaration section of a non-OO program). Once the specification is incorporated, the next thing to do is to tell the database when to use this new module. It is done by creating a new set of response or modifying an existing set of response. In the relational table of the relational database, this is nothing more than putting in an additional set of entries to the database.

The term "multi-thread" refers to the different selections of control algorithms as shown in Figure 11. Assuming the performance of different control algorithms varies on different processes and different optimization level of a process, fuzzy logic can play a role here to determine which algorithms to be used. At this point, the multi-thread idea is a proposal. A lot of factors have to be considered in terms of establishing a logic to choose which algorithm to be used. For example, some algorithms tend to build up the accuracy as the run number increases. Whether a given process has that much room to improve is also another question.

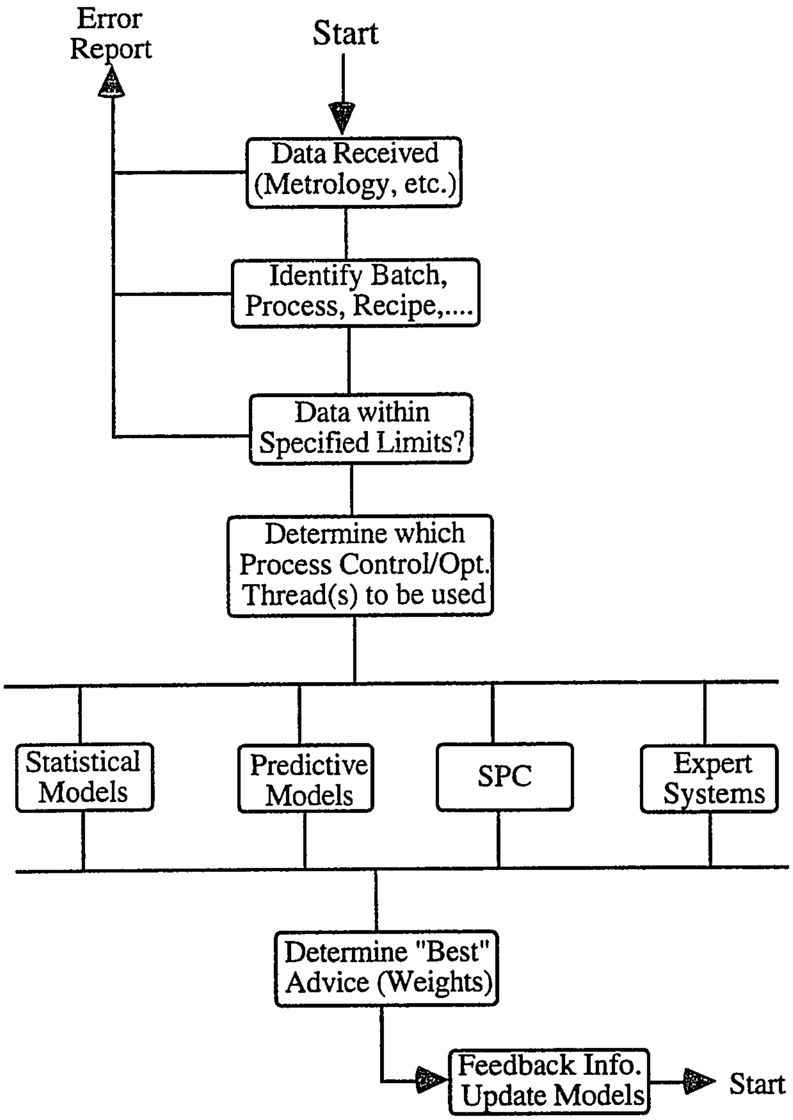
Error Report
Start
Data Received (Metrology, etc.)
Identify Batch, Process, Recipe,....
Data within Specified Limits?
Determine which Process Control/Opt. Thread(s) to be used
Statistical Models
Predictive Models
SPC
Expert Systems
Determine "Best" Advice (Weights)
Feedback Info. Update Models
Start

Figure 11: The process optimization and control module with multiple threads

An implementation of the GCC is in progress for the optimization and control of a plasma etching process in an Applied Materials 8300 Reactive Ion Etcher. Figure 12 shows the configuration of the control system. The plasma controller is a real-time controller that attempts to hold the control parameters at the Plasma Generation Subsystem given by the run-to-run controller. It is a very common control configuration for utilizing both real-time and run-to-run controllers.

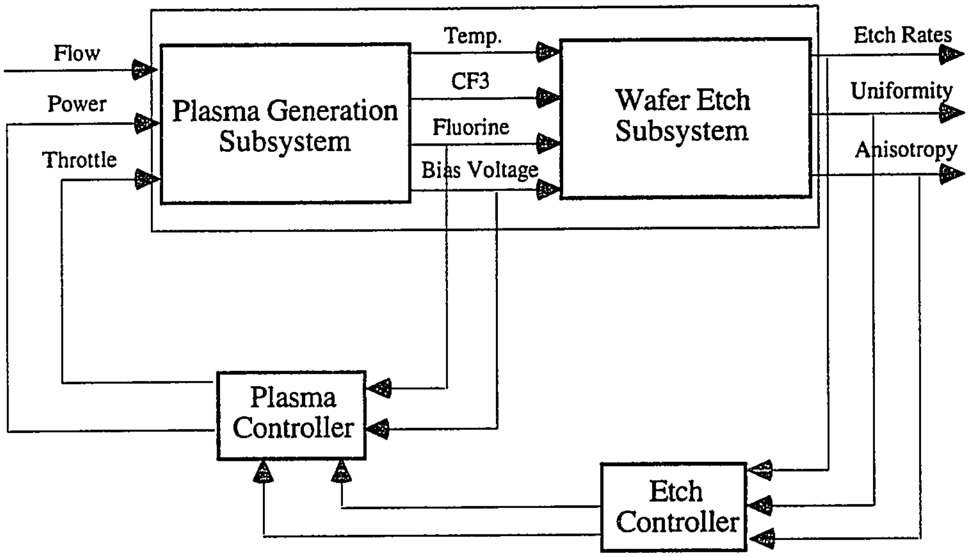

Figure 12: Close-loop Control System for Plasma Etching

A report of the work at the University of Michigan is documented in Appendix IV.

## 2.5. THE RUN-BY-RUN PROCESS CONTROL SYSTEM OF MIT FOR VLSI FABRICATION

The RbR ( Run-by-Run ) process control system is being developed in a academic research environment. It is a modular process control framework for VLSI fabrication. The goal of the system is to enable higher yields on both small and large lots of wafers. The system consists of three core modules: the flexible recipe generator, the RbR controller, and the real-time controller.

The structure of the system is shown in Figure 13. Based on the mask geometry and specifications of each new product design, the flexible recipe generator module provides a recipe that brings the process into the general region of the best performance at the first run [13]. This global optimization scheme seeks the region of process space that is most robust against disturbances.

Starting from the recipe provided by the flexible recipe generator module, the RbR Controller updates the recipe between each run of the fabrication process. Based on the post-process measurements, the RbR Controller updates the process models and recommends a change in the recipe to compensate for the drifts and shifts of the process. Figure 14 shows the structure of the RbR Controller [14]. The RbR Controller consists of three modules: the generalized SPC, rapid mode and gradual mode. The generalized SPC is a diagnostic module that decides when to use the rapid mode module. The purpose of rapid mode is to rapidly compensate for shifts to the process that are caused by, for example, specification changes and maintenance operation. The gradual mode module is to compensate for drifts in the process that are caused by noise [14]. The latest application of

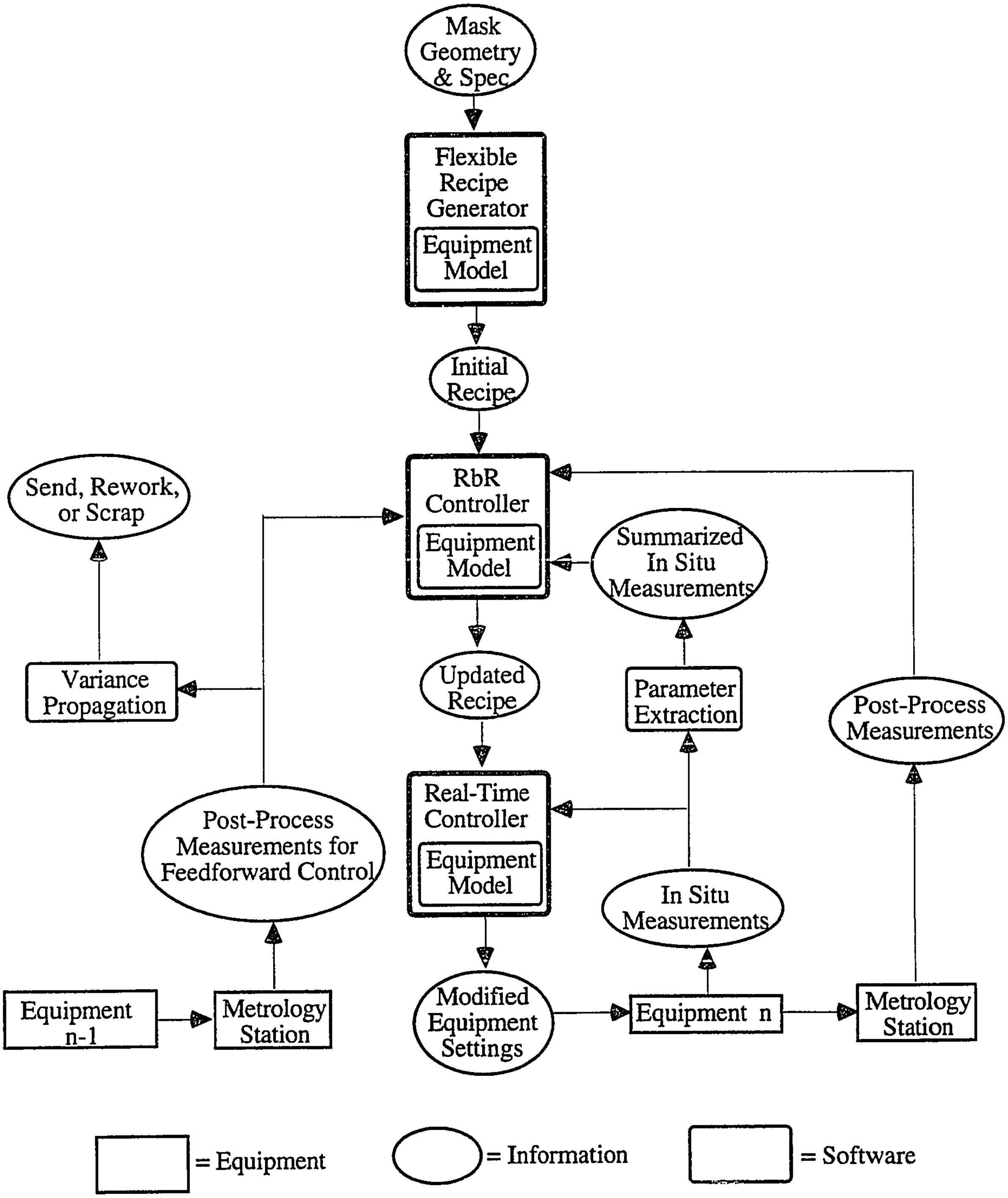

Figure 13: Block diagram of process control system for VLSI fabrication.

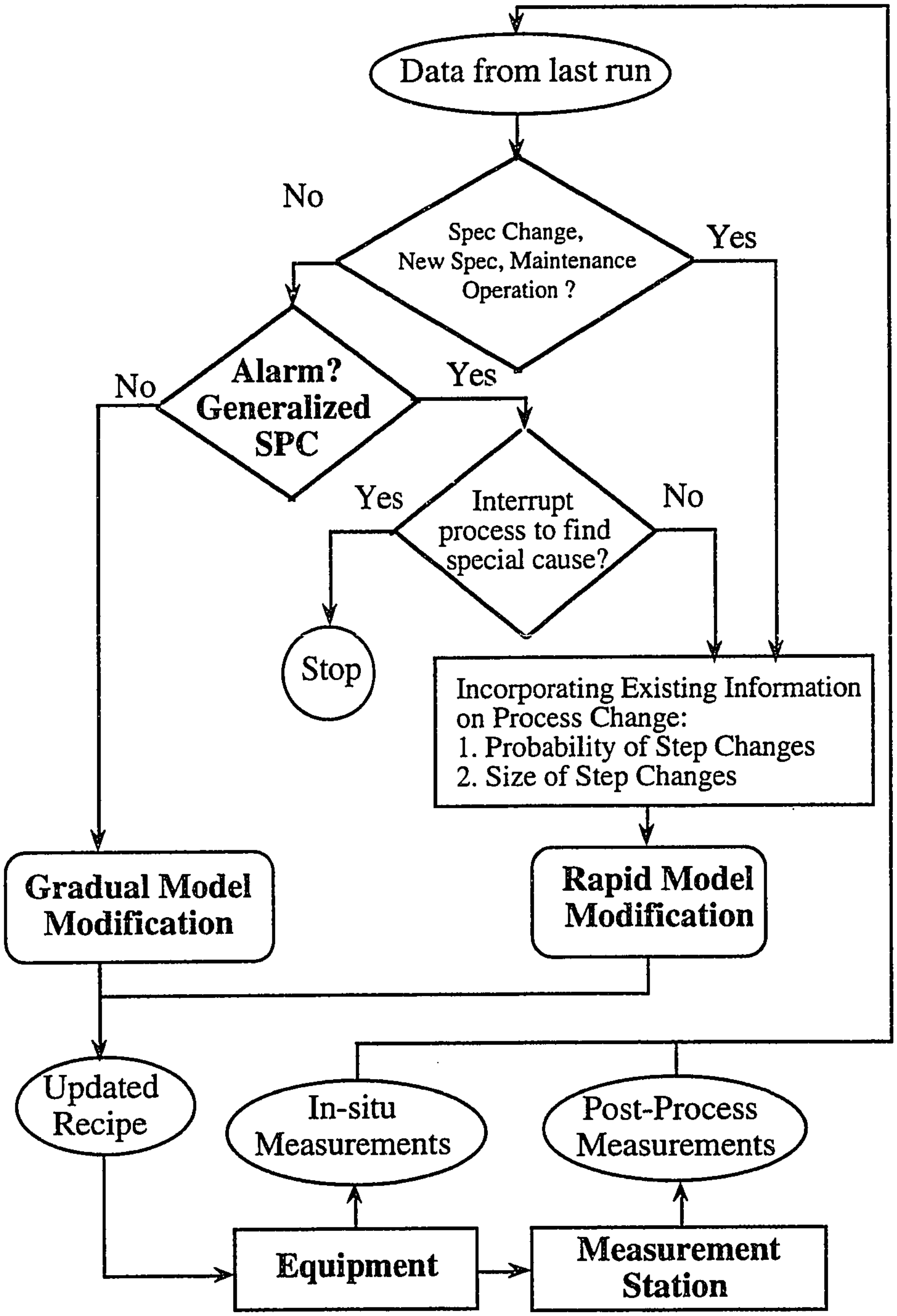
Data from last run
No
Spec Change, New Spec, Maintenance Operation ?
Yes
No
Alarm? Generalized SPC
Yes
Yes
Interrupt process to find special cause?
No
Stop
Incorporating Existing Information on Process Change:
1. Probability of Step Changes
2. Size of Step Changes
Gradual Model Modification
Rapid Model Modification
Updated Recipe
In-situ Measurements
Post-Process Measurements
Equipment
Measurement Station

Figure 14: The block diagram of the Run by Run Controller. It has three major algorithmic modules: the generalized SPC module, the rapid mode module and the gradual mode module. It also has two branch points: one on whether the current run occurs right after a specification change or maintenance operation, the other on whether to interrupt the process to identify and correct the special cause.

the RbR Controller to CMP process is recorded in chapter 3 of this thesis

During a process step, the real-time controller module modifies the equipment settings based on the *in situ* measurements. In order to accomplish this, equipment models that include time is used to relate the equipment settings to *in situ* measurements of process parameters.

## 2.6. BERKELEY COMPUTER AIDED MANUFACTURING SYSTEM

The Berkeley computer aided manufacturing System (BCAM) is an object-oriented framework that aims to support all aspects of equipment and process control in semiconductor manufacturing. BCAM is a part of a CIM project, whose objective is to develop and implement a prototype of a future semiconductor manufacturing plant. C++ and Common Lisp Object System (CLOS) are the primary development tools. X windows are used in the monitoring, SPC and diagnostic modules. BCAM is a UNIX workstation based system. It consists of six modules. They are the real time monitoring module, real time SPC module, equipment maintenance record keeping module, real-time monitoring and real-time SPC, automated diagnosis of equipment condition, fault diagnosis, recipe

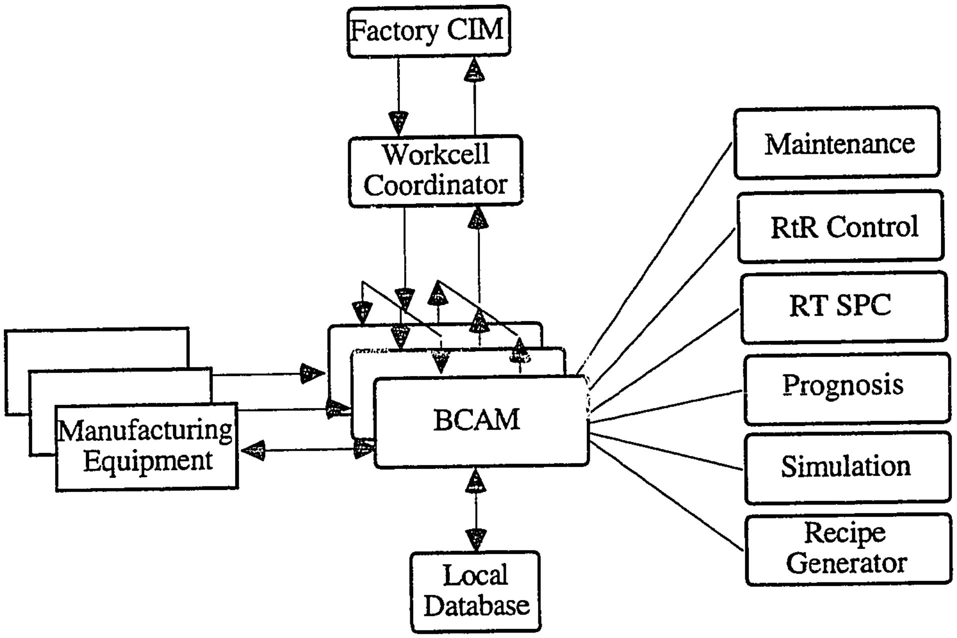

Figure 15: BCAM Architecture

generator and model simulation module. Also, BCAM provides generic utilities, such as communication server, graphical user interface, numerical and statistical libraries as external resources [15].

Figure 15 shows the architecture of BCAM. Notice that the level under the cell coordinator can be expanded to incorporate multiple cells with BCAM. The RTSPC (real-time SPC) system inspects a process by applying SPC principles on data that is collected through a real-time sensor via the SECS II interface. The data collection process is fully automated, and an unsupervised time-series model generation feature is also included in the system. The RTSPC system will be one of the commercialized process and equipment control application software programs in the near future.

# CHAPTER 3: CIM SYSTEMS IMPLEMENTATION

Three CIM system implementation projects are reported in the following sections. They are the Run-by-Run (RbR) Control of the Chemical Mechanical Planarization Process (CMP) at National Semiconductor, the FABCON from Japan and the Adaptable Manufacturing System (AMS) of the cluster-tool fab at Stanford University. The implementation of the RbR control of the CMP process is currently in progress. The FABCON is shown here to illustrate an implementation of CIM system that is relatively low cost, and yet sophisticated and highly automated. The AMS is shown here to illustrate the performance of AMS over the traditional Mass Manufacturing System (MMS).

## 3.1. RUN-BY-RUN CONTROL OF THE CHEMICAL MECHANICAL PLANARIZATION PROCESS IN NSC

The RbR Controller [16] is a model-based control system which provides a framework for controlling semiconductor manufacturing processes subject to disturbances such as shifts and drifts as a normal part of their operation. The RbR Controller has been applied [17][18] successfully to a technically matured epitaxy process at AT&T Microelectronics, Allentown, PA, and has demonstrated major improvements over the results from standard process control methods in the fab.

However, the development of the process models for model-based controller can be time consuming and expensive if done on-site, interrupting the production. To overcome this problem, the RbR Controller is being deployed concurrently on the CMP process in development at Fairchild Research Center at National Semiconductor Corporation (NSC). The concurrent deployment of the RbR Controller during the CMP development phase

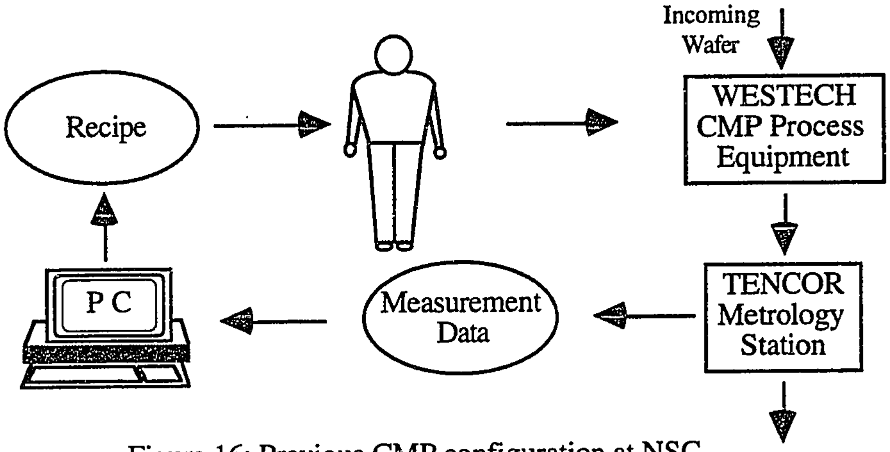

Figure 16: Previous CMP configuration at NSC

includes development of the process model, testing RbR control of the process, and building a cell controller to establish the interface between process equipment, metrology station and fab operators.

New CMP process equipment (Westech) and metrology station (Tencor) have been installed. The connection between the Tencor metrology station and the PC has been established previously as shown in Figure 16. The metrology station and the PC shared the same computer hard disk for data storage. An equipment interface program written in C and SDR (SECS II driver from GW Associates ) is used to handle the communication between the PC and the Tencor metrology station. The PC contained an algorithm written in Matlab to retrieve the measurement data, perform analysis and generate a updated recipe. The recipe was then loaded into the CMP equipment by the operator. The process model was developed on the new CMP process equipment, and it will be used for RbR Control.

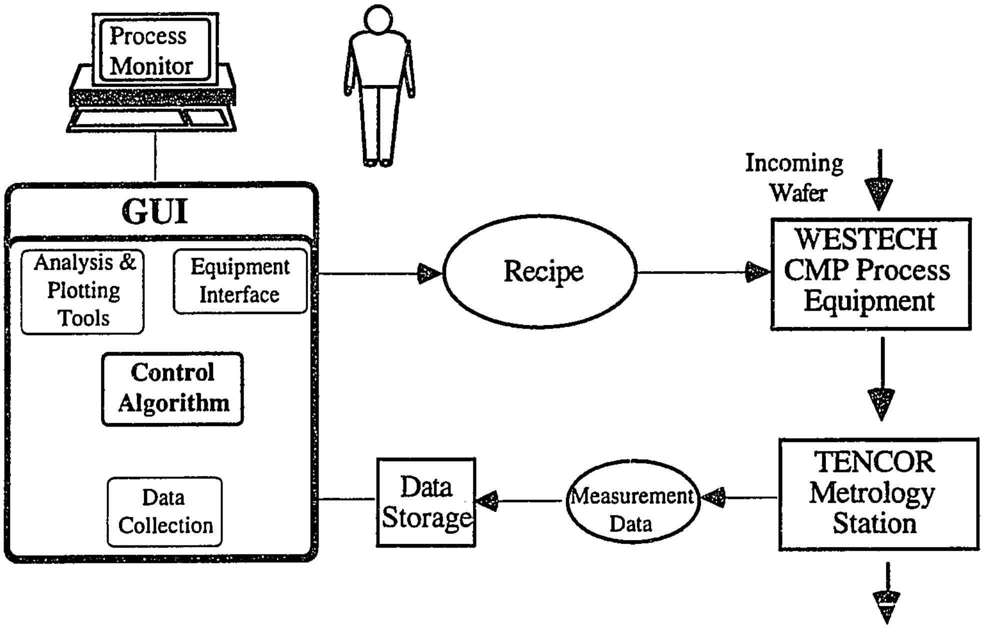

Figure 17: Current configuration of the CMP cell controller

The current configuration of the cell controller at Fairchild Research Center is illustrated in Figure 17. Visual Basic ( from Microsoft ) is used to code the control algorithm of the cell controller. The cell controller will consists of a graphical user interface, data collection routines, equipment interface routines and the control algorithm.

A gradual mode multivariate RbR Controller (MVRbR) is currently being developed and tested at San Jose State University. The formulation of the MVRbR is documented in Appendix III. An EWMA algorithm is used for the gradual mode. This algorithm will be used to control the CMP process at the Fairchild Research Center. The CMP process will be controlled by using the EWMA algorithm of the gradual mode will be used along with the pad age effect, a compensation function that simulates the decade of the polishing pad effectiveness obtained from a process characterization experiment. The software structure of the MVRbR Controller is shown in Figure 18. Currently, the algorithm is being implemented using Matlab.

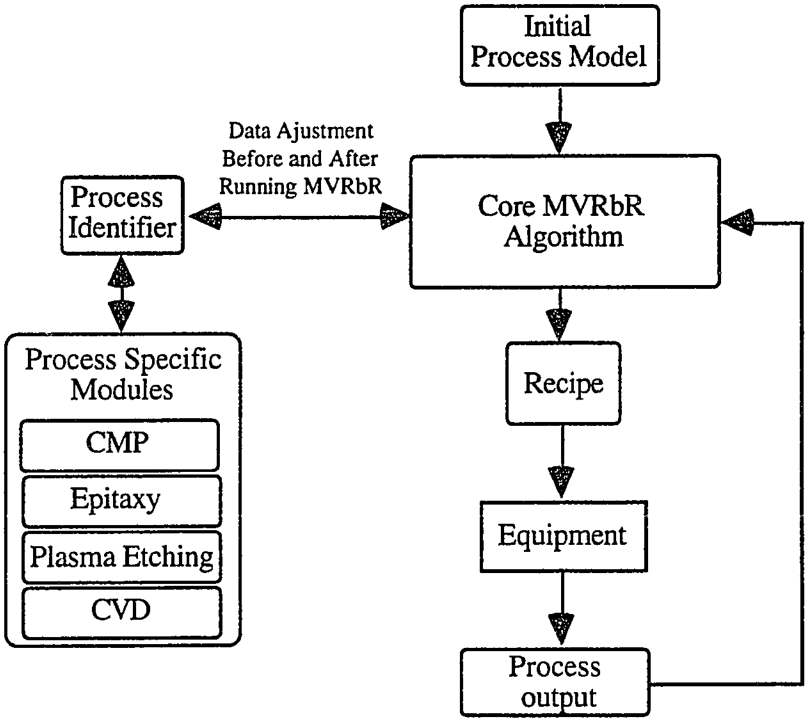

Figure 18: Software Structure of the Multivariate RbR Controller

The current version of the MISO (Multiple Inputs Single Output) RbR Controller is 1.2. The changes in this version enable the RbR Controller to run on PC and UNIX Matlab 4.0. Also, the rapid mode algorithm of the controller has been modified so that it can function in situation where only rapid mode is needed. The detail changes are recorded in Appendix II of this paper.

## 3.2. FABCON

An automation system called FABCON has been developed and implemented in an ASIC fab as a joint venture between LSI Logic Corporation of U.S. and Kawasaki Steel Corporation of Japan. Along with the traditional CIM functions of automatic lot tracking, recipe setting, and process data collection, this system also creates the production schedule automatically, and controls the transport of almost all wafer carriers throughout the fab by robots.

All communication between FABCON and the process and measurement equipment is based on the SECS standard. For the robot-equipment communication, optical PIO interface units and an interlock communication protocol were used for interlock control between inter-bay robots and equipment.

FABCON architecture is inherently distributed. The host factory MIS system manages planning, inventory, and factory reporting. The distributed bay control system provides lot tracking, equipment control and monitoring, fab workload balancing and real-time scheduling. Bay control is hosted by a network of PCs, running the FABCON modules. These modules run on top of the QNX operating system. QNX is a micro-kernel operating system particularly suited to real-time distributed systems [19].

Similar to SCC, factory growth can be incrementally supported as shown in Figure 19. As new bays are brought on-line, additional nodes can be added to the overall control system with almost no software change. Also, PC based computer systems offer sufficient speed and reliability in FABCON. Table 2 [19] shows the hardware configuration of the system

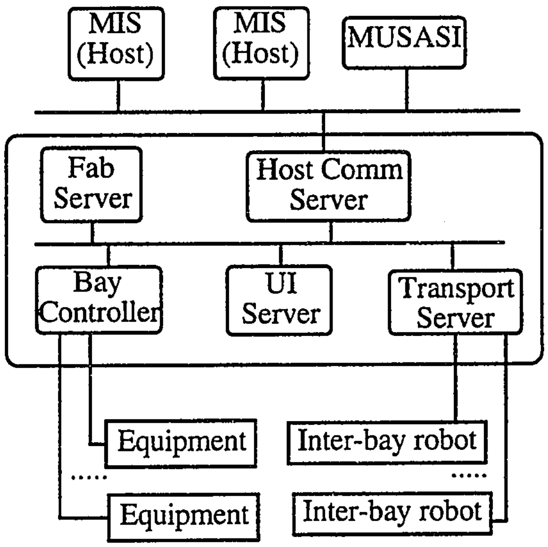

Figure 19: FABCON Architecture

| **Module** | **PC** |
|---|---|
| Fab Server | 486/50 with 600MB HD, 16 MB memory |
| Bay Controller | 386/33 with 110MB HD, 16MB memory |
| User Interface Server | 386/33 with 110 MB HD, 8MB memory |
| Transport Server | 486/50 with 110 MB HD, 16MB memory |
| Host Comm. Server | 386/33 with 110 MB HD, 8MB memory |

Table 2: Hardware Configuration of FABCON modules.

## 3.3. THE ADAPTABLE MANUFACTURING SYSTEM FOR CLUSTER-BASED FAB

Product cost has been the only dominant measure of fab's economic performance for the past ten years, which led to a primary focus on utilization and yield. Although they are still crucial to a fab's profitability and productivity, the ability to produce various products, time to market and capital cost are now of comparable importance to product cost. An alternative approach, Adaptable Manufacturing System (AMS), has emerged in response to these changing requirements on semiconductor manufacturing. The goal of AMS is to optimize the cost versus throughput time tradeoff. Comparing this new approach with the conventional approach, Mass Manufacturing System (MMS) has more emphasis on minimizing cost per wafer.

The current conception of AMS at Stanford University is based heavily on the Texas Instruments' MMST configuration. It is because of the significant contribution of single-wafer processing technologies by the MMST program. By replacing slower batch equipment with single-wafer processors, the throughput time of the entire process flow can be reduced in AMS fab. Except for the high current implanter, all AMS equipment is single-wafer and most are arranged into clusters of three or four process modules. The current AMS model uses only dry processing. The dry wafer cleaning technologies that are used in the AMS model are based on a number of recent publications by groups, including those a Penn State and Fujitsu. However, the simulation results have shown that an AMS fab could use wet processes in stand-alone or clustered single-wafer wet tools without significant degradation of its performance.

Closed-loop real-time process control must be used in order to make AMS feasible. During the pilot stage, it is a competitive advantage to have the ability to quickly specify a process to a machine, and then to have the machine accurately run the process. Also, in

| | **IC MMS** | **AMS Fab** |
|---|---|---|
| Fraction Single-Wafer Processes | 60% | 98% |
| Typical Throughput Time | 3 weeks | 6 days |
| Dominant Equipment Configuration | stand-alone | clustered |
| Process Control | open-loop, in-line SPC | closed-loop, real-time |
| Process Metrology | stand-alone | in-situ |
| Dominant Equipment Architecture | closed | open |
| Lot Size | 12 to 48 wafers | 1 to 24 wafers |
| Efficient Capacity (wafers/month) | 10,000 to 25,000 | 5,000 to 20,000 |
| Number of Products | <10 | 1000s |
| Typical WIP Inventory (wafers) | 10,000 to 50,000 | 1,000 to 10,000 |
| Product Type | commodity | value-added and commodity |
| Wafer Carriers | standard cassettes | vacuum sealed cassettes |
| 1-wafer minimum cycle time | 181.3 hrs | 37.8 hrs |
| 24-wafer minimum cycle time | 283.0 hrs | 140.0 hrs |

Table 3: Conception of a Mass Manufacturing System and Adaptable Manufacturing System.

order to be able to fabricate a wide variety of products, the ability to switch process equipment reliably from one process to another is critical. Furthermore, closed-looped process control can maintain yields by enabling more robust processes, and by detecting equipment problems earlier than in an ex-situ process. Based mainly on the experience at Stanford University 's AMS simulation and in MMST program, the modeled cost of in-situ process monitors and software control would increase the total cost of most process modules by up to 10%. Table 3 [20] shows some quantitative and qualitative conception of a MMS and an AMS configuration.

The performance of AMS throughput is a result of a number of emerging enabling technologies. There are basically three broad technological changes in process technology enabling the AMS: the ability to replace almost all batch processors with single-wafer processors, being able to cluster multiple processors without significant loss of process control, and the reduction of setup times [20].

# CHAPTER 4: DISCUSSIONS AND CONCLUSION

## 4.1 DISCUSSIONS

As the number of the process steps increases, the number of processes to be designed and executed within a single facility increases accordingly. The use of information technology becomes essential in this situation. A computer-readable and computer-manipulable representation of process can allow the use of such information technology so that anyone in the fab can access information pertinent to his/her work in a common and easily understand format. In this respect, the SCC's Baseline Scenario, the MMST's logical architecture, and CAFE's process flow representation serve the purpose of having a common process structure that can be understood and carried out in the CIM system.

Unlike the industrial CIM systems such as the MMST or SCC, equipment interface is not the main focus of CAFE. CAFE does not know when a process is started or stopped because there is no direct interface between machines and CAFE (except a Nanospec metrology instrument in the fab, which has a single thread interface with CAFE that can obtain measurement data automatically). The only bridge between CAFE and the machines is the engineer who works in the fab. When a job starts, engineers would enter the starting time into CAFE through the lab terminals in the fab. When a job has finished, engineers would enter the time, results of the lot and additional comments into the CAFE system. In the fab, CAFE acts as a fab record keeping and scheduling system. Through the inputs from the engineers, it keeps track of the status of each wafer lot and records scientific data. Although CAFE seems to lack efficiency in terms of automation and equipment interface, it is appropriate for research and development fabs that require flexibility to do experiments. A CIM system that has delicate equipment interface for automation would be more suitable for some industrial fabs that need to avoid human errors and maintain fabrication consistency.

In contrast to other industrial client server environments such as Strategic Cell Controller (SCC) of SEMATECH, the main focus for CAFE's client server environment is in the fab. Client server environment in other CIM systems usually involves sharing of databases, applications ( development software ), and CPU. CAFE has more focus on sharing the database and CPU. Moreover the sharing of database and CPU concentrates in the fab area. CAFE itself runs as a one huge program that stands alone. Although application is one of the menu items of the CAFE system, there are no other development software applications being integrated into CAFE. Development is being done outside the scope of CAFE. Integrating an application into a CIM system is not a simple matter. In an industrial environment, it is more important to integrate applications into the CIM to reduce the cost of incompatibility. For example, SGI's CIM system contains one single circuit CAD tool that is being used throughout the entire company. Designs can be downloaded to the manufacturing line within minutes. In a research and development environment, this kind of speed might not be necessary.

In terms of process representation, PFR strongly emphasizes the declarative specification of information about a process and minimizes the manipulative and computational flexibility accessible within the PFR. Interpreters have the responsibility for the manipulation of PFR objects. Fabrication methods are supported only within an interpreter, and not within the PFR, which consists of only simple functions. This approach is different from that of some other process representations, where the user specifies operation data objects and methods [8].

In terms of implementation methodology, SCC develops standards along with the CIM system and its different modules, whereas MMST develops the CIM system before the standardization is fully considered. Developing standards and the CIM system together seems to be a logical approach at the first glance. However, the progress of the development on two issues can interfere with each other and affect the overall development of the program. Utilizing existing standards can speed up the development of the CIM system and its different modules. With less concern for commercial standardization, more new ideas can be incorporated into the development without additional obstacles, and this is the implementation methodology that is used by the MMST program. First, Smalltalk, an object oriented equipment interface, was used at the development stage. After the development had come to a satisfactory stage, a process control software named ControlWORKS based on the MMST program was developed for the industry, in which the software is generic enough to fit different equipment interfaces and different CIM systems.

The development of the software industry is much faster than the development of the semiconductor fabrication technology. Often, software being used would have gone through a several upgrades during one design and implementation cycle of a new technology. A fully developed new technology might be implemented through outdated software, which can no longer be used to communicate with the newly developed CIM system. Or, a fully developed new technology is implemented through the latest software that the engineer who developed the new technology is not familiar with. Software implementation is a big burden for the implementation of a new technology. To reduce the occurrence of software implementation failure, one possible solution is to develop the new technology and the related software concurrently. This solution makes the MMST program stand out among many other CIM system developers. MMST program concurrently develops software, processing equipment, processing techniques, process control, equipment control and sensors. The gap between software implementation and technological development is greatly reduced.

The semiconductor manufacturing industry may be too young (compared with other industries) to support a dedicated software supplier base. As an alternative, software from other industries is widely used by process and equipment control engineers. However, these software packages, which are not designed for this industry, require enormous amounts of customization effort from the engineers simply because of the complexity and instability of semiconductor manufacturing processes. Moreover, because of the varieties of software available in the market, different groups of engineers within a facility often use different software packages for their design and analysis. As the process and equipment control technology grows, reusability of the non-standardized and highly customized software is low. Consequently, the software environment in the semiconductor manufacturing systems is highly fragmented, and process and equipment integration become even more difficult. The fragmentation is being accelerated by the fact that the semiconductor manufacturing industry lacks a common software development platform and the support from the fast growth of software development.

Instead of treating process and equipment control technologies as proprietary products, companies such as TI have started commercializing their technologies. As a result, there are some emerging process and equipment control software packages, such as ControlWORKS and RTSPC, that are specifically designed for the semiconductor manufacturing industry. These software packages are usually ready-to-use. Engineers can avoid repetitive effort in terms of implementing control algorithms into software coding by utilizing these software packages as process and equipment control modules.

The philosophy of adaptability behind SCC, ATP and GCC is applicable in terms of solving the fragmentation problem in the software environment of the industry. Fragmented software can be integrated into one single platform using an object-oriented approach. A standard should be established for the software developer so that individual software can be treated as an object or module. Notice that object-oriented and the traditional function/data software development approach are different. Software developed using object-oriented approach has proven to be more adaptable and less time consuming in terms of development. Treating individual application software as an independent object, using the object-oriented development approach, allows the system to be developed in modules. The integration of different modules can be simplified by establishing a common platform that can supervise and communicate with each module. Since the growth of process and equipment control is much faster than the shop floor control, this common platform should first be targeted to implement at the cell level.

## 4.2 CONCLUSION

From the studies of different equipment and process control CIM system for semiconductor fabrication, the author has the following suggestions for future CIM system developers:

- Object-Oriented design approach: This approach would enhance software adaptability and expandability. Currently, C++ and Smalltalk are the most popular object-oriented design tools.
- Use of existing software product: Save time and effort in process and equipment control software development. Avoid re-inventing existing technology. Software packages such as TI's ControlWORKs, provide lower level equipment control as well as the higher level user interface developer.
- Use of relational database system: Relational database should be used to systematically record the dynamic data (data about the current system status) and static data (initial values or knowledge-base type of instructions) of a CIM system. This would provide a frame work for programmers to obtain information or instruction of the system using Standard Query Language.
- Messaging protocol: Before choosing a messaging protocol, developers should consider carefully about the capability of networking (outside the CIM system) and the expandability (within the CIM systems client-server frame work).
- Cluster tool: Cluster tool based fab is considered as fab of the future. In equipment control, CIM systems developers need to construct the system so that multiple process chambers of a cluster tool can be controlled through one control module. This is quite different than one control module controlling one equipment. The processing sequence of jobs in cluster tool makes the control logic more complex.

# REFERENCES

[1] Rod Buchanan," Equipment Control Definitions and Strategies: a Tutorial," Proceeding of SEMATECH AEC/APC Workshop, Dallas, Texas, October 1993.

[2] "SIA Semiconductor Technology - Workshop Conclusions," Semiconductor Industry Association, 1993.

[3] "SIA Semiconductor Technology - Workshop Working Group Reports," Semiconductor Industry Association, 1993.

[4] Kevin Nguyen, " The Development and Implementation of SEMATECH Strategic Cell Controller ( SCC ) Framework," Proceeding of the Fifth Annual International Semiconductor Manufacturing Science Symposium ( ISMSS 93 ), San Francisco, California, July 1993.

[5] Kevin Nguyen, " A Common Graphical User Interface Framework for Equipment and Process Control," Proceeding of SEMATECH AEC/APC Workshop, Dallas, Texas, October 1993.

[6] John McGehee, John Hebley and Mike Pfauth, " The MMST Computer Integrated Manufacturing System: an Overview (sic)," Texas Instruments Technical Journal, vol.9, no.5, pp.87-98, September-October 1992.

[7] Duane S. Boning, " Semiconductor Process Design: Representations, Tools, and Methodologies," Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, January 1991.

[8] Rajeev Jayavant, " A User/Programmer Guide to the Fabform User Interface," VLSI Memo No. 88-487, Microsystems Research Center, MIT, Cambridge, MA, November 1988.

[9] M. Sullivan, S. W. Butler, J. Hirsch, and C. J. Wang, "A Control-to-Target Architecture for Process Control," IEEE Trans. Semiconductor Manufacturing, vol 7, no. 2, pp.134-147, May, 1994.

[10] R. Beaver, A. Coleman, D. Draheim, and A. Hoffman, "Architecture and Overview of MMST Machine Control," IEEE Trans. Semiconductor Manufacturing, vol 7, no. 2, pp.127-133, May, 1994.

[11] Andy Hoffman, "Design Approach to ControlWORKS," Proceeding of SEMATECH AEC/APC Workshop, Dallas, Texas, October 1993.

[12] Moyne, J.R., H. Etemad, and M. Elta. " A Run-to-Run Control Framework for VLSI Manufacturing," SPIE - Microelectronics Processing, Monterey, California, 1993.

[13] A. Hu, E. Sachs, R.S. Guo, and S. Ha, " Process Control System for VLSI Fabrication," IEEE Transactions on Semiconductor Manufacturing, vol. 4, no. 2, May 1991.

[14] A. Hu, E. Sachs, A. Ingolfsson, and P. Langer, "Process Control of an Epitaxial Silicon Deposition Process," Sixth SRC/DARPA CIM-IC Conference, Raleigh, NC, Aug. 1991.

[15] N.H. Chang, " Monitoring, Maintenance and Diagnosis in a Computer-Integrated Environment for Semiconductor Manufacturing," Electronics Research Laboratory, University of California, Berkeley, CA, July 1990.

[16] E. Sachs, A. Hu, and A. Ingolfsson, "Run By Run Process Control: Combining SPC and Feedback Control," accepted for publication, IEEE Transactions on Semiconductor Manufacturing.

[17] A. Hu, E. Sachs, A. Ingolfsson, and P. Langer, " Run by Run Process Control: Performance Benchmarks," Proceeding of the Fourth Annual International Semiconductor Manufacturing Science Symposium (ISMSS 92), IEEE Cat No. 92 CH 3161-7, San Francisco, CA, June 1992.

[18] A. Hu, " An Optimal Bayesian Process Controller for Flexible Manufacturing Processes," Department of Mechanical Engineering, MIT, Cambridge, MA, August 1992.

[19] Akira Matsuyama and Jessi Niou, " A State-of-the-art Automation System of an ASIC Wafer FAb in Japan," Proceeding of the Fifth Annual International Semiconductor Manufacturing Science Symposium ( ISMSS 93 ), San Francisco, California, July 1993.

[20] Sameuel C. Wood and Krishna C. Saraswat, " Configuration and Management Strategies for Cluster-Based Fabs," Proceeding of the Fifth Annual International Semiconductor Manufacturing Science Symposium ( ISMSS 93 ), San Francisco, California, July 1993.

# APPENDIX I: MIT CAFE SYSTEM

## A. HARDWARE & SOFTWARE REQUIREMENTS

- Sun O.S. 4.1.3

- 48M of RAM minimum. MIT has 128M in the CAFE machine. Thomas Lohman recommended 64M of RAM.
  Approximately 16M of RAM will be taken by the database.

- 400M of disk space minimum is required for CAFE and Ingres. This is not including the disk space for the O.S. and the Xwindow system. 600M of total disk space will be needed to accommodate the O.S.(140M approx.) and the Xwindow (50M approx.). MIT currently has two 1G external disk drive for the CAFE machine. Thomas Lohman recommended 1G of disk space.

- Ask for the large generic Kernel from Sun. The large generic Kernel will have everything CAFE needs. No modification is required on the Kernel in order to get CAFE running.

- For simplicity, do not ask for the multi-processor architecture version of the Sun. MIT currently is using the single processor version of a Sun SPARC 10.

- Ingres 6.4/03.

- The following menus from Ingres are needed: Installation & Operation Guide, Database Administration Guide, Ingres/Net and Embedded Quel/SQL.
  Ingres/Net is a menu specifically for setting up a client server environment. Assuming that more than one workstation is going to share the Ingres database, this menu will explain what the setup for Ingres will be.
  Embedded Quel/SQL do not come with the standard package but we must ask for it.

## B. INSTALLATION OF CAFE SYSTEM

- Installation sequence:

1. The Sun O.S. 4.1.3
2. Ingres Database
3. Xwindow
4. Gestalt & Schema
5. CAFE

- The Sun O.S. will be installed by Sun. The installation of Ingres should follow the Ingres Installation menu. The X11 library can be loaded from the public domain of MIT and copy into our machine.

- Thomas Lohman will use the following routines to make a compressed "tar" version of all the source code and directories and ftp to San Jose State U:

tar-CAFE, tar-source and distrib-source.

These c routines are in the /CAFE directory of the CAFE machine in MIT. What they do is to make a copy a CAFE release from the development CAFE tree. The routine tar-source search through directory tree and tar all source code ( including symbolic links, which only points to different directories on the same workstation in this case ) and released copies of RCS files ( RCS is a revision control system which documented all the revisions, reason for revising and the corresponding corrections ) up into one single compressed file. The routine distrib-tree will create a compressed version of the source directory tree from the development source tree. This will allow us to build the source directory tree and the corresponding source codes.

- Installation of GESTALT

A higher-level view of the procedure:

1. Building the core system. This complies all the Gestalt source code, packaging the result in the form of a library.

2. Build test database. This simply provides a database, which is required before any programs may be run. ( This step also builds a C interface to the test database )

3. Testing the DBTEST database. There are two test programs in the DBTEST database, predef and test_db. The programs are self-checking and report any errors with a message.

4. Creating additional GESTALT database ( the real one).

A step-by-step 6-page-menu written by Thomas Lohman contains details of the procedure above.

The above procedure included creating the minimal schema for the GESTALT to function normally. However, if we have a schema source file at the time of installation ( the most likely case ), we can put that file into the load_schema.c file under the misc subdirectory in the database's root source directory. This will allow a complete schema to be built.

• To install CAFE, MIT will provide a compressed version of CAFE directory tree and source code. The directory tree should be " un-tar " first. Then " un-tar " the source code itself.

• The previous procedure is very straight forward. Although CAFE contains a lot of source code files, the process should take less than a day to finish. The main concern is the "factory " setup of our side - SJSU. According to Thomas Lohman and Greg Fischer, none of the data ( data such as facility, machine, operation sets, process flow and specifications that are unique to MIT ) will be transferred to us. There is an unique machine, nanospec, that CAFE will look for. We must make changes to the corresponding pfr files to avoid error. All these files are within a directory called /CAFE/pfr/opsets. In order to run our own CAFE system, we have to set up our own " factory," which has machine, process flow and operation sets. To avoid problems at the beginning, the pattern of the " factory " should follow the one in MIT.

• George Young is the person who installed CAFE at Lincoln recently. A good reference source besides Greg and Thomas. The following are the reference sources in terms of installing and tuning the CAFE:

| | | | |
|---|---|---|---|
| George Young | gry@mtl.mit.edu | | |
| Greg Fischer | fischer@mtl.mit.edu | MIT Room 39-315 | 617-253-3371 |
| Thomas Lohman | thomasl@mtl.mit.edu | MIT Room 36-287 | 617-258-6485 |

For RbR controller integration, contact the following person for the latest progress:

Mihir mihir@athena.mit.edu or mihir@goesser.mit.edu or mihir@mtl.mit.edu

## C. MECHANISM OF CAFE

There is more than one way to utilize CAFE. The following is an example to show one of the ways that starts a process using a process flow representation file (pfr). Also, the sequences below are not unique. Other combinations are possible. It is meant to be a sample tour of the CAFE system from both an user and software developer point of view.

• A process flow representation file should first be available before going into CAFE and start a process.

- pfr file is a file written in Common Lisp. User either creates a pfr file using an editor (UNIX vi is one of the possibilities) or modify an existing one. Although pfr files are written in Lisp, they are easy enough to be read for someone who does not know Lisp. They contains a process flow name and a series of pre-defined operation names. These operations are actually subroutine calls which contain information for creating a process flow tree. A process flow tree is nothing more than a series of sub-processes. To create a process, one must first get familiar with the baseline processes of CAFE system. The baseline processes are the standard building blocks of process flow. Next section will discuss baseline in more detail.

∘ After CAFE is invoked, specify a facility at the facility field on the menu screen. It is not a requirement to put in the facility first. However, knowing that machine is a subset of facility, it is a logical approach to select the facility first. The name of the facility can either be typed manually or selected from the scrolling choices (Cntrl A).

- The menu itself is created by running the Lisp source codes ( Initialize.lisp, Input-screen.lisp ) in /usr/CAFE/src directory. Not only the format and text arrangement are done by the Lisp source codes, but also the features of going from one menu to another (the layering of the menus). The scrolling feature at some of the fields on the menu is made possible by a routine called one-of, which restricts the choices of a specific field. The choices come from the database.

• Go to Process Flow menu, then go to Install Process menu. Input the process name. Note that the inputting process name must be identical to the one at the first line of the process flow file. Input the user name. For the PFR Filename field, the actual path should also be specified. Press Cntrl C to save or Cntrl X to quit without save.

- CAFE runs the pfr file according to the information specified at the PFR Filename field. The subroutine within the pfr file will be invoked. A process tree will be created by a Lisp code called PROCESSFLOW.lisp. Also process description will be created by another Lisp code called PROCESSDESCRIPTION.lisp. These are called schema. We now have schema called processflow and processdescription. Tools that allow us to visualize schema will be introduced later.

• After the pfr file has been loaded, we need to create a wafer lot to manufacture using the processes specified in the pfr file. Hit space bar to return to the top level menus. Go to Lot Management menu. Then go to create a lot menu. The User name field can be filled in by hitting return. A lot name can also be assigned by hitting return at the lot name field. At the Process name field, again choices can be scrolled by pressing Cntrl A. Number of wafers, separate wafer sets and numbering can also be specified on this menu. Press Cnrl C to save or Cntrl X to quit without save.

- This step creates another schema called process instance. This is being done by invoking a routine called PROCESSINSTANCE.lisp. Note that the restricted scrolling choices at the processing name field are the ones that are installed at the Install Process menu.

• Right after a lot is created, CAFE will ask for the wafer IDs and types. By leaving the laser ID fields blank, the ID will be name as 'Wn' where n is an integer. For the Wafer Type fields, CAFE has a sets of restricted types to choose from. Cntrl C will save the entry and Cntrl X will quit without saving. After the verification of the wafer laser IDs and types, CAFE will ask you to create a wafer set. If wafer set is needed, type a wafer set name and select the wafers that you want to be in this set by changing the field next to the wafer ID from No to Yes. After these entries have been saved, CAFE will ask if additional wafer set is needed. If so, create another set as before. If not, CAFE will bring back the Lot Management menus.

• At the Lot Management menus, we can now tell CAFE which lot we would like to work with. Move the cursor to the Lot field and select the lot names that have been created earlier by pressing Cntrl A. Now the lot can be processed by selecting Start Lot from the menu.

CAFE will then list a set of operations which will be done based on the process that has been selected earlier.

- CAFE creates a Task tree from the process description flowroot tree by invoking TASK.lisp and LOT.lisp. Each task under the Task tree has corresponding flow objects form the process flow tree. There are only three types of tasks in CAFE - planned, active or done.

• Now, go to the Fabrication Tools menu. Type in N for the next operation to be done on the wafer lot. Then select a machine name, and put in the starting date and time from the scheduling screen. After the operation is finished, the ending date and time should be entered. Also, comment, machine settings and resulting reading ( if any ) on that particular operation can be entered as well.

- Normally, when the operation performs in the fab, information will be entered from the fab ( unless it is only a demo ). CAFE is not a real time system. Engineers who perform the operations will be responsible for putting in information to the CAFE system after the operation has finished. The information usually includes date, time, comments ( in text form, can be very detail ), machine setting, the recipe code and the result reading (if any). There is another computer that stores all the recipes. This computer is located in the fab and is a stand-alone unit. Engineer usually obtains recipes from that computer and performs the operation. The name ( or code ) of the recipe will then be entered into the CAFE system as a record. Through the Task editor, the recipe being used on every lot can be found.
- At the Fabrication Tools menu, typing in N (for the next operation) triggers the nextop.lisp subroutine. It searches through the process flow tree and finds out what the next operation is. Also, choosing " Operate a Machine " from the menu will trigger the operate-machine.lisp subroutine. Based on the current operation, operate-machine will find out what are the machines that can be used to complete this step.

• After a Lot has been created, task tree will be created. Through the Task Editor, the process flow and operation status can be viewed in a graphical manner. Clicking on the tasks will show the sub-tasks windows. Further and further clicking will eventually bring up the processes and what has been done on each wafer.

- This is a good tool to visualize what the schema looks like. So far, the following schema have been created: PROCESSDESCRIPTION, PROCESS FLOW, LOT, PROCESSINSTANCE, and TASK. All of these objects are created through GESTALT using schema routines and located at the Ingres database. This graphical tool allows us to look at the hierarchy of the task tree and how one object relates to another. The mouse clicking and pop-up window feature is very user friendly. Although CAFE is an ASCII driven system at the current stage, CAFE will be further developed toward a more graphical user interface.

• All the transactions are recorded in the database. Documentation of the transactions can be obtained from the Traveler Report menu. It will give the status of each operation in the process flow for the lot.

The above procedure is an example of how to run the CAFE system. Besides the user interface, it also shows the software action that runs behind the user interface. Basically, users need to have a pfr file, which is a process planning file written in Lisp that outlines what needs to be done on the wafer. After this has been done, the user would tell CAFE to apply these processes to a specific set of wafers. This can be done by selecting an existing wafer lot or creating a new one. Therefore, pfr file and wafer lot are the basic ingredients in terms of running the CAFE system. Base on the information given by the pfr file, CAFE would be able to find the corresponding information, such as choices of machines that can do the operation, which lab would have such machine, the corresponding operations of each process and sequence of the operations. This is the result of the formation of the corresponding schema (install process) - processdescription, process flow, lot, processinstance and task will be created, which can be considered as objects of the database. A process flow tree will then be formed, and the entire job will follow the sequence of this process flow tree.

## D. SOFTWARE STRUCTURE OF CAFE

CAFE directory tree is very big. It is very difficult to map out and explain the functionality of the entire directory tree. But there are some source code and files that play important roles in CAFE system. Looking into these source code and files allows us to understand more about the structure of CAFE system. Although some of these source code or files are unique to MIT (might not be usable in SJSU), the structure and logic will still applicable to most general cases. Knowing more about these source code or files will help the " tuning " of our own CAFE system as well.

- Baseline is defined as unique information of factory and manufacturing at MIT's labs. It defines sequence of processes, the operations sequence under each process, which machine should run which operation, machine settings, which recipe to be used, time required, written instruction. Basically, the hierarchy can be defined as follows: process flow, task, process, operation and specifications. Baseline defines the relationship from process to specifications. The following paragraphs briefly explain how this is arranged in terms of coding. Again, this unique information can only be used in MIT. SJSU will have to create its own factory. However, the same methodology of defining process flow, task, process, operation and specifications should be used.

- Directory /CAFE/pfr/opsets/opset_flow/baseline contains the " baseline " of the following process: diff, etch, implant, metal and photo. " baseline " is understood as the detail content of each process. For example, under /CAFE/pfr/opsets/opset_flow/baseline/diff, there are about 13 diffusion process files ( their names are the name of diffusion processes ). In each file, there are a series of *.fl call, which call the *.fl files from /CAFE/pfr/lib.

- Files under the directory /CAFE/pfr/opsets/opset_pfr/baseline have the same name as the files in /CAFE/pfr/opsets/opset_flow/baseline. However, the content in the files is different. These files do not have routine calls; rather, they have the specifications such as which machine to be used, which recipe to be used, written instruction of the process, time required, etc. Files under this directory are being called by the files in /CAFE/pfr/opsets/opset_flow/baseline. This information is unique to each process. With this information, engineers in the fab can actually set up their machines and run the job. Using the task editor (still under development, tasks can only be viewed at this point) of the CAFE system, the hierarchy of opset_flow and opset_pfrs can be visualized. The information from the files in this directory can also be seen from the tasks editor.

- Under /CAFE/include, a lisp code fabform.h contains all the externally visible definitions for subroutine callable version of Fabform. Looking into this file, one can find out what are the current subroutines that are being used for Fabform.

• Under /CAFE/include, a lisp code eqpmt-util.h contains the definitions of various equipment utility routines. This file also explains the effects and requirements of each equipment utility routine, and provides an overview of all the equipment routines.

• Any file in the directory /CAFE/include/gdm with a .lisp extension is a database object. By looking into these files, you can see what slots ( smaller objects ) are in each database object and what function you can use to access the slots.

• schema_dump is an executable located in /CAFE/bin. It will list out the attributes of a database object. For example, schema_dump task will print the attributes of a TASK database object. Using this along with list the gdm files, one can find out what objects are available in the database and what database objects point to what other objects.

# APPENDIX II: CHANGES IN THE VERSION 1.2 OF RBR CONTROLLER RUNNING IN MATLAB 4.0 FOR WINDOWS 3.1 AND UNIX SYSTEMS

- **Change file names** - The data file name of the RbR controller must not exceed 8 characters. Also, the suffix of the file names must not exceed 3 characters. For example, hardmat.hard is a data file in the workstation version. This is not a valid file name to be called in the PC version. hardmat.har would be a valid name. Another example is file_util.m; it should be changed to a 8 character file name such as file_uti.m. The file calling lines in the beginning portion of the Matlab file rbr.m should be changed according to the new file names

- **Check file format** - All data files must be in ASCII format. Transferring workstation ASCII data file into PC sometimes distorts the format of the data file. In that case, reading error will occur. Modify the data files when necessary.

- **Specify the path** - Before running RbR controller in Matlab, the path that specifies the location of the related files must be specified. Using the change directory command to move to the current directory will not solve the problem, because the file calling lines tell Matlab to search for files according to the path. For example, if all the rbr files are stored in c:/matlab/toolbox/matlab/rbr, you must type in path(path,'c:/matlab/toolbox/matlab/rbr) at the Matlab command window before running rbr.m

- **Continuation statements** - Generally, if the command lines are too long, they can be separated into 2 lines by typing in 2 dots ( .. ). The PC version of Matlab does not like the 2 dots, it prefers 3 dots ( ... ) or more. There are quite a few 2 dots being used in the entire program. The best way to solve this problem is to use a editor that has a find/change option to search through all the programs.

- **Error trap** - Currently, an error trap in onerun.m ( starting in line 247 ) is being disabled. The error trap is to check the size of the matrix for the inputting data. However, there were no lines before that error trap to initialize the dimension of that matrix. Error occurs when it tries to read in the dimension of the matrix at the first loop. The second loop would have been running fine since data is received starting from the second loop. The same problem should occur in the workstation version, but the fact is that it works fine in the workstation version. This problem is still being investigated. However, this problem does not affect the RbR algorithm at all. It runs safely unless the user does not follow the directions and put in less data than the program expected.

- **one command** - The usage of the one() command has been changed. To obtain an identity matrix that has the same dimension as, say matrix x, the command is one(size(x)) in the latest version of Matlab, as opposed to one(x). This will generate a number of incorrect matrix dimension error.

- **Ployline is outdated** - In graph.m, ployline is being used to plot the graphs. However, this command is outdated in this version of Matlab. Matlab can still activate this old algorithm but a message saying this command is outdated appears on the screen every time ployline is used. The equivalent command for this version would be the line command.

- **File utility** - The file utility of the RbR Controller did not function properly due to the loading and saving command changes in Matlab. All files without a subscript .mat are considered to be ASCII files. The matrices to be saved using the file utility are not in ASCII format. The subscripts in the file utility have been changed to .mat.

- **The New Logic for the Rapid Mode** - In previous versions, the rapid mode would only engage in situations where the generalized SPC alarm was on. If rapid mode was used with the gradual mode off, the recipe would stay the same until the alarm was triggered. Therefore, new logic is required for running the rapid mode alone. Three scenarios are considered here:

  **a.** When both the gradual mode and the rapid mode are on, alarm is required in order to determine whether the rapid mode should be used. Therefore, alarm dependency is required in this scenario.

  **b.** When gradual mode is off and the rapid mode is on, alarm is not required because rapid mode is the only algorithm to be used. Therefore, there is no alarm dependency in this scenario.

c. When gradual mode is on and the rapid mode is off, alarm is not required because gradual mode is the only algorithm to be used. Therefore, this is an alarm independent scenario.

Based on the above three scenario, the following logical statement is used to determine whether the rapid mode is needed:

```
if ( not(gradual) and rapid ) or (alarm and rapid and gradual) then
    find the step location and run the rapid mode
end
```

The above expression is implemented in one of the RbR Matlab M-files called onerun.m. The table below illustrates the logic of the expression above

| Scenario | Gradual | Rapid | Alarm | Logic | Result |
|---|---|---|---|---|---|
| a | 1 | 0 | 1 | (0&0)\|(1&0&1) | 0 |
| a | 1 | 0 | 0 | (0&0)\|(0&0&1) | 0 |
| b | 0 | 1 | 1 | (1&1)\|(1&1&0) | 1 |
| b | 0 | 1 | 0 | (1&1)\|(0&1&0) | 1 |
| c | 1 | 1 | 1 | (0&1)\|(1&1&1) | 1 |
| c | 1 | 1 | 0 | (0&1)\|(0&1&1) | 0 |

◦ **Starting with Rapid Mode at the First Run** - In previous versions, size and location of step change can not be determined at the first run because the algorithm expected a series of data points ( or a single data point ) from the process before the step occurs. Without the initial data points ( or point ), the algorithm could not make any estimates of step location or step size. For the RbR Controller to recognize the step change, a set of initial target values should be incorporated and have the Controller do the comparison starting at the first run. This set of initial target values should be input by the user or generated from the process model without adding any noise. The only condition that requires such an adjustment is in the case of running rapid mode starting from the first run. The following expressions are used in the algorithm to generate the initial values for the comparison:

```
if ( not(gradual) and rapid and (run =1) ) then
    set the first row of the corresponding matrices to the target values
```

```
        misc(7) = 1
    end
    if ( misc(7) = 1 )
        increase the run number by 1
    end
```

The above expressions are implemented in one of the RbR Matlab M-files called onerun.m. Also, a global boolean variable is added to the rbr.m file as an indicator of using rapid mode from the first run. This variable is the seventh element of the matrix "misc". Note that the target values stay in the matrices. For the rest of the runs, only the run number is modified so that the sizes of the matrices are consistent.

# APPENDIX III: MULTIVARIATE RBR CONTROL ALGORITHM

## Multivariate RbR Control Algorithm: Gradual mode

The predicted output is given by: $\hat{Y}_t = BX_t + A_t$ ---------- (1)
where the predicted ouput $\hat{Y}_t$ is { m x 1}, the slope terms $B$ of the process model is {m x n}, the recipe $X_t$ is{n x 1} and the constant terms $A_t$ of the process model is {m x 1}. In gradual mode, the constant term is updated using EWMA algorithm as follows:

$A_t = \Lambda(Y - BX_{t-1}) + (I - \Lambda)A_{t-1}$ ---------- (2)

where Y is the actual output from the process, and $\Lambda$ is a {m x m} EWMA weight matrix. To solve for $X_t$, the following condition must be satisfied:

$T = BX_t + A_t$ ---------- (3)

In (3), T is the target value of the process. Depending on the number of inputs and the number of output of the process model, the problem boils down into three cases.

<u>**Exact case**</u>: m = n

In this case, enough information is given to solve equation (3) above. The solution of $X_t$ is given as follows:

$X_t = B^{-1}(T - A_t)$ ---------- (4)

Note that $X_t$ here is the exact solution of equation (3). Therefore the predicted ouput $\hat{Y}_t$ is always equal to the target value T.

<u>**Overdetermined case**</u>: m > n

In this case, too much information is given to solve for equation (3). So, there are infinite numbers of solutions. The following expression would give a unique solution to equation (3):

$$X_t = \left(B^T B\right)^{-1} B^T (T - A_t) \quad \text{----------} (5)$$

Note that equation (5) only gives a $X_t$ that is least square fitted to $\left(T - A_t\right)$ and $B$. Therefore, it is not an exact solution to equation (3). So, the predicted output $\hat{Y}_t$ is always a certain distance away from the target value T.

**<u>Underdetermined case</u>**: m < n

In this case, not enough information is given to solve equation (3) above. Therefore, additional information is needed in order to solve $X_t$. Here we would like to find a $X_t$ that is closest to $X_{t-1}$ such that $\left|X_t - X_{t-1}\right|^2$ is minimized. This implies the following:

Cost Function to be minimized: $f(X_t) = X_t^T K X_{t-1} - 2X_{t-1}^T X_t + X_{t-1}^T X_{t-1}$

Subject to an equality constraint: $h(X_t) = BX_t - (T - A_t)$

The Lagrange conditions are:

$$\nabla f(X_t) + \lambda \nabla h(X_t) = 0$$
$$h(X_t) = 0$$

Assuming K is positive definite,

$$2KX_t - 2X_{t-1} + B^T\lambda = 0 \quad \text{----------} (6)$$
$$BX_t - (T - A_t) = 0 \quad \text{----------} (7)$$

From (6),

$$2KX_t = 2X_{t-1} - B^T\lambda$$

$$X_t = K^{-1}(X_{t-1} - \frac{1}{2}B^T\lambda) \quad \text{----------} \ (8)$$

Substitute (8) into (7),

$$BK^{-1}(X_{t-1} - \frac{1}{2}B^T\lambda) - (T - A_t) = 0$$
$$BK^{-1}X_{t-1} - \frac{1}{2}BK^{-1}B^T\lambda - (T - A_t) = 0$$
$$\frac{1}{2}BK^{-1}B^T\lambda = BK^{-1}X_{t-1} + (T - A_t)$$

$$\lambda = 2\left(BK^{-1}B^T\right)^{-1}\left[BK^{-1}X_{t-1} + (T - A_t)\right] \quad \text{----------} \ (9)$$

Substitute (9) into (8),

$$\begin{aligned} X_t &= K^{-1}\left\{X_{t-1} - B^T\left(BK^{-1}B^T\right)^{-1}\left[BK^{-1}X_{t-1} + (T - A_t)\right]\right\} \\ &= K^{-1}X_{t-1} - K^{-1}B^T\left(BK^{-1}B^T\right)^{-1}BK^{-1}X_{t-1} + K^{-1}B^T\left(BK^{-1}B^T\right)^{-1}(T - A_t) \end{aligned}$$

The solution of $X_t$ can be expressed in the following form:

$$X_t = K^{-1}\left[I - B^T\left(BK^{-1}B^T\right)^{-1}BK^{-1}\right]X_{t-1} + K^{-1}B^T\left(BK^{-1}B^T\right)^{-1}(T - A_t) \quad \text{----------} \ (10)$$

For $K = I$, expression (10) can be reduced to

$$X_t = \left[I - B^T\left(BB^T\right)^{-1}B\right]X_{t-1} + B^T\left(BB^T\right)^{-1}(T - A_t) \quad \text{----------} \ (11)$$

# APPENDIX IV: THE UNIVERSITY OF MICHIGAN CONFERENCE REPORT

## GCC

- Currently, GCC is being developed on NEXT computers. Object C is being used in the coding. Object C is a much more object-oriented programming language than c++. It can be viewed as a combination of Smalltalk and c++. The NEXT platform is capable of generating executables that can be run on DOS machines, which serve the purpose of the project since the deliverables are going to be loaded into a PC broad. Also, NEXT is Sun compatible, which means anything that can run on Sun can run on NEXT.

- Nauman is responsible for the database development and Roland is responsible for the overall development of GCC.

- There are basically six objects in the GCC framework: modules, message (un)parser, I/O gateway, conductor, timer and database. The term "object" in here is the object in the object-oriented design framework. They communicate with each other using a messaging format, which is a standard procedure in object-oriented programming (message is sent simply by pointing an arrow to another object; for example, cout<<"output " in c++, would send the word "output" to an object called cout). Note that the modules, which are objects in the GCC framework, do not communicate with other modules directly. Modules are in passive positions. They would not be executed unless the database tell them to do so. First, the database would accept a request from the user. Then the database would search for an appropriate response, which includes the execution sequence of the individual modules and the corresponding data or setting that goes with the module. Then the conductor takes the inquires from the database and executes individual modules in the sequence that is provided by the database. Each module would acknowledge the conductor once the modules have finished their tasks. Then the conductor would execute the next module and so on.

- Communication between individual modules and the equipment is established through the message parser and the I/O gateway. The modules sends out a c++ message to the message parser. Then the message is forwarded to the I/O gateway which handles all the TCP/IP networking communication to the "outside" world. However, the individual modules can be written in such a way that they communicate with equipment directly without going through the message parser and the I/O gateway. This could apply in real-time control.

- The database has two sides, the knowledge base and system state data (in individual data file format). The knowledge base stores the information about the actions on a given message. When a message comes in, the database search for the corresponding action items. The action items then translated into specific names of routines, and the corresponding parameters that associate with the routine. Then, this message that consists of the names of the routine and the corresponding arguments in order will be sent to the conductor so that the ordered routines can be executed in sequence.

- This is called an expert system, which consists of a knowledge base, an inference engine and working memory. The knowledge base defines the rules on how to manipulate and analyze a given set of data that is in the working memory. The manipulation is done through the inference engine. In GCC, the knowledge base is represented by the relational database system, the working memory is the data that generates by the processes, and the inference engine is represented by the different modules.

- In order to put additional modules into GCC, one must follow a set of specifications. This set of specifications is nothing more than the definition of an object. To be more precise, the specification is the class definition in object-oriented programming (similar to the variable and/or type declaration section of a non-OO program). Once the specification is incorporated, the next thing to do is to tell the database when to use this new module. It is done by creating a new set of response or modifying an existing set of response. In the relational table of the relational database, this is nothing more than putting in an additional set of entries to the database.

- Currently, the development of GCC is in a stage of "integration." Since different students are responsible for different tasks, they have come to a point where they need to put their portion together.

- The demonstration was done on the NEXT computer. Mainly, the user interface was demonstrated. Xwindows is being used for the user interface. Also, Ultramax is currently integrated into the GCC.

## Multi-thread Architecture of GCC

- Nauman is responsible for this development

- The term "multi-thread" refers to the different selections of control algorithms. Assuming the performance of different control algorithms varies on different processes and different optimization levels of a process, fuzzy logic can play a role here to determine which algorithms to use.

- Currently, this is a proposal. A lot of factors have to be considered before an algorithm is chosen. For example, some algorithms tend to build up the accuracy as the run number increases. Whether a given process has that much room to improve is also another question.

## Suggestions on the CIM Systems Overview

- According to James, we have picked the major players in the overview. One other possibility is the G2 system of GENSYN. They have not heard much about the AMS system of Stanford.

- CAFE seems to be a much higher level system that handles supervision and scheduling. The rest of the systems in the overview seem to be much more involved in process and equipment control.

- Despite the fact that BCAM seems to have quite a lot of accomplishment, Arnon said that their deliverables were not easy to implement in the past. The industry has not gained much from the BCAM program.

- Controlworks: they would like to know more details about it, e.g., examples of applications, their development tools, the plug-and-play capability, what do they use specifically in their process control modules.

- ATP: they would like to know the plug-and-play specification for software vendors.

- BCAM: they would like to know how generic it is, and the plug-and-play capability

- In the software development side, why would they choose their development environment and tools? Have they conducted any in-depth study on the effects?

- Application partitioning: it is a good idea; however, it has to be done dynamically in order to have a truly flexible set up.

- Reusability and OOD: this is quite an involved question and is very difficult to evaluate.

- Memory paging and caching method that they use.

## Real-time Control Update

- Pramod Khargonekar is responsible for the development. No handouts were available after the presentation, but James promised that he will mail us one.

- Real-time control is applied in plasma etching. The system consists of two control loops, one is the inner loop which is the real-time control loop that control the flow rate and the temperature within the chamber, the other one is the outer loop which is the Run to Run control loop that control the wafer to wafer etch rate.

- Note that at this stage only the real-time control loop is being implemented. The Run to Run control loop has not been accomplished yet. But it will be their next tasks.

- Two example were shown, one is the etch rate using close loop real-time control versus etch rate without using close loop real-time control under normal circumstance; the other one is the stability of etch rate using close loop real-time control versus the stability of etch rate without using closed loop real-time control in a situation where the gas pump is leaking.

## Note

- In order to integrate our new RbR algorithm into the GCC, it will be easier if we modify their existing R2R code (suggested by James). The main reason is that the code has already incorporated the specification to integrate with GCC.