A Process-Independent Run-to-Run Controller and Its Application to Chemical-Mechanical Planarization

James R. Moyne, Roland Telfeyan
University of Michigan
Display Technology Mfg. Center
Ann Arbor, MI 48109-2108
Phone: (313) 936-3645
email: moyne@eecs.umich.edu

Arnon Hurwitz
Statistical Methods and Advanced Process Control
SEMATECH
2706 Montopolis Dr., Austin, TX 78741
Phone: (512) 356-3177
email: Arnon.Hurwitz@sematech.org

John Taylor
Compugenesis, Inc.
10605 Walpole Lane., Austin, TX 78739
Phone: (512)288-9376
email: jtaylor@compugen.com

Abstract – A process-independent run-to-run (R2R) controller has been developed and successfully applied to the control of a chemical-mechanical planarization (CMP) process. The controller design utilizes a Generic Cell Controller (GCC) enabler; thus it is process-independent, and the controller implementation software exhibits a high degree of portability, flexibility, robustness, and re-usability. The design also includes a multi-branch R2R control scheme that can incorporate any number controller algorithms in a complementary fashion. Further, it provides support for data collection, R2R recipe optimization and control, and recipe advice download. The controller implementation is largely hardware and software independent; its operation has been demonstrated on SUN SPARC, Intel 486 and Pentium, and HP PA-RISC platforms. It has a capability to incorporate in dynamic fashion (i.e., during run-time) any number of software modules existing on any of the aforementioned platforms within a distributed environment, resulting in a truly dynamic and distributed solution. The implementation was initially applied to the control of a reactive ion etcher (RIE). More recently it has also been successfully applied to the R2R control of a CMP tool, thus demonstrating process independence. The latter application utilizes a "gradual mode" MIMO linear approximation control algorithm developed at MIT, enhanced to support parameter weighting and advice parameter granularity. Recent results indicate that good control of removal rate with fair control of uniformity has been achieved. Current efforts are focused on development of additional algorithm "branches" to complement the gradual mode control, and on the reduction of process variance through real-time equipment monitoring.

I. Introduction

Run-to-run (R2R) control is a form of discrete process and machine control in which the product recipe with respect to a particular machine process is modified ex-situ, i.e., between machine "runs", so as to minimize process drift, shift, and variability. This type of control is a critical component of the hierarchical scheme that is widely suggested for facility control in the semiconductor manufacturing arena [1, 2].

A review of the state-of-the-art in this area reveals that there are a number of algorithms available to address aspects of R2R control [3-9], yet their utilization in semiconductor manufacturing facilities has been slow to occur. The primary reason for this lack of acceptance and utilization is that, until recently, there was no framework for R2R control that allowed for controller software portability and reuse, adaptability, and robustness. Also, there was no generalized mechanism for testing and verifying (on-line) the effectiveness and domain of applicability of algorithms. Further, there was no accepted process-independent procedure or template for the development and deployment of R2R control.

During the past few years a research effort has been focused on addressing these issues so that R2R control may be better utilized in the industry [1, 10]. The main goal of this effort has been to develop and demonstrate a process-independent R2R controller. This paper contains a discussion of results of this effort to-date. Indeed the most significant result is that a process-independent run-to-run controller has been developed and demonstrated. The controller is enabled through the use of Generic Cell Controller (GCC) technology; the GCC approach provides for a controller that is process and platform independent, adaptable, robust, and portable (i.e., re-usable) [11, 12]. The use of the controller has been demonstrated in the control of plasma etching and more recently chemical-mechanical planarization (CMP) processes [12].

This paper is organized as follows. Background material on the GCC R2R control enabling mechanism as well as the CMP process is presented in Section 2. Section 3 details the design requirements for a R2R controller in the industry. The GCC-enabled R2R controller that meets these requirements is described in Section 4; this description focuses on a software implementation of the R2R controller that has been utilized in the control of etching and CMP processes. Section 5 addresses selected features of this implementation in more detail, while the utilization of the controller in the control of a CMP process is addressed in Section 6. This document concludes with a brief summary and a discussion of future directions in further controller development.

II. Background

The Generic Cell Controller

The GCC is a discrete control mechanism that utilizes a relational database as opposed to procedural code to store the sequential control information of a controller. The theory of operation of the GCC is documented in the literature [11, 12]. The main feature of the GCC that makes it an attractive R2R control enabler is that the database schema is tailored for the storage of event driven sequences that dictate how the system is respond to events; these sequences are stored as data in the database. For this reason the GCC is capable of enabling complex and dynamic control scenarios that are characteristic of many R2R control systems. Further, due to GCC database schema and interaction specifications, a very high degree of modularity is established with GCC applications. This results in both high portability and transferability of software, and a capability to easily incorporate commercially available software components into the system.

A schematic illustrating the use of the GCC in a typical R2R control implementation, in this case CMP control, is given in Figure 1 [1, 10]. The GCC provides for intelligent routing between the various software modules involved in the R2R control task; these modules may include commercially available software elements such as communications drivers and controller algorithms. Note that the GCC provides a venue for the comparative evaluation of optimization and control algorithms, as it can incorporate any number of these algorithms and provide sequences for their selective utilization. The algorithm evaluation could include the investigation of paradigms for the complementary utilization of a number of control algorithms to achieve more robust control [1, 12].

Related results associated with the development and deployment of the GCC implementation described in this paper have also been documented in literature. Specifically, practical issues associated with the utilization of a R2R control algorithm (see Figure 1) are discussed in [3]. Issues of R2R control that relate specifically to CMP are discussed in [13]. A detailed discussion of the operation of the current GCC in providing CMP process R2R control is provided in [10].

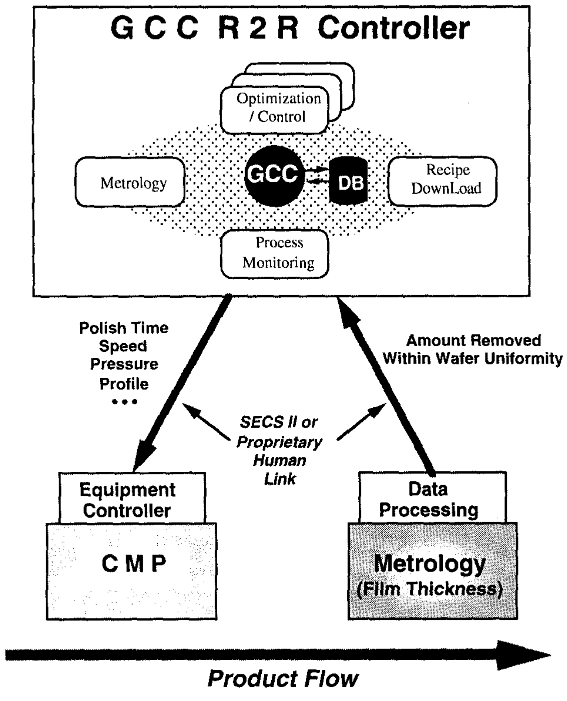


Figure 1: GCC Utilization in R2R Control: Example CMP Control

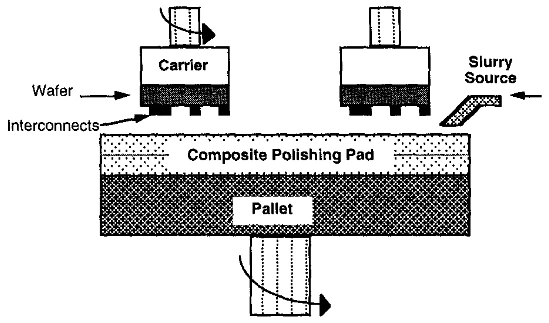
Chemical-Mechanical Planarization

Chemical-Mechanical Planarization (or polishing) has become a widely accepted technology for multilevel interconnects. CMP of dielectric films is the planarization method of choice for 0.35-μm device technology. In addition to providing planarization, CMP has also been shown to reduce defect density and define vertical and horizontal wiring [14].

CMP is basically a surface planarization method in which a wafer is affixed to a carrier and pressed face-down on a rotating platen holding a polishing pad (see Figure 2). A silica-based alkaline slurry is applied during polishing thus providing a chemical and mechanical component to the polishing process. The general process goal is the preferential removal of "high" material across the wafer. Typical process metrics include removal rate (or amount removed) and within-wafer-uniformity. Equipment and process parameters that are typically utilized to control the process include polish time, pressure, rotation speed, and parameters that impact the conditioning of the polishing pad such as conditioning profile.

There are a number of characteristics of CMP that make it an ideal candidate for the development, implementation and test of R2R control. First, the process is not well understood. This combined with factors such as inconsistency and degradation of consumables, and lack of sensors and actuators makes CMP a challenging candidate for control. Second, as there is a lack of in-situ sensors for CMP, in-situ control is not yet feasible; thus R2R control appears to be the tightest form of control that can be applied to CMP at this time.

The CMP process has been described in much greater detail elsewhere in the literature, notably [14, 15]. The CMP R2R control problem is detailed in [10, 13]. A summary of the state-of-the-art of CMP utilization including a discussion of limitations of the process and its control is presented in [16]. A second effort focused on the development of CMP R2R control is described in [8, 9].



Inputs	Outputs
Pressure	Removal Rate
Polishing Time	Within Wafer Non-Uniformity
Temperature	Wafer-to-Wafer Non-Uniformity
Platen Velocity	BullsEye
Carrier Velocity	Endpoint
Platten Velocity Conditioning	Over/Under Polishing
Conditioning Time	Memory Effects
•••	•••

Figure 2: Typical CMP Process

III. Design Requirements

The low level of acceptance of R2R technology in the semiconductor manufacturing arena indicates that adequate R2R controller design requirements have not been met for the painless integration of industrial quality R2R control. Clearly the primary concern of integrators is the short and long term cost of developing, integrating, utilizing, and maintaining systems utilizing R2R control. However, it is the issues affecting cost that dictate the design of the controller.

In order to understand the issues that impact the cost of R2R control it is necessary to describe the technical and practical semiconductor manufacturing R2R control environment.

First and foremost, semiconductor processes are complex, not well understood (thus physical models are generally non-predictive), and very dynamic. Further the control technology applied to these processes is basically sensor driven, and there are generally an insufficient number of sensors and actuators at each semiconductor manufacturing process step to establish a desired level of control over process parameters. Thus there is a tendency to rely on empirical methods for control. A number of empirical control algorithms have been developed for application to R2R control in semiconductor manufacturing [3-9]. Algorithm implementations and other supporting software elements (communications drivers, user interface modules, etc.) have been made available from a number of sources, both commercial and non-commercial. However, in general, the domain of applicability of each algorithm is limited or not well understood. A design requirement for a R2R controller thus is that it must be able to incorporate the multitude of available software elements and utilize them in complementary fashion as necessary so as to achieve adaptable and robust control.

Providing this level of integration implies a number of design requirements for the R2R control implementation. It must be able to incorporate third-party (commercial) software such as the algorithm and supporting software elements just described. It must provide a mechanism for the easy integration of these software elements. It must be hardware and software independent and distributed so as to utilize these elements running in their native environments. It must be portable to a multitude of software and hardware environments, and must be as process-independent as possible, so as to maximize reusability. Additionally, the solution must be realizable with off-the-shelf components so that industrial-quality system development and maintenance can be achieved. In summary, the solution must provide generic, process-independent, flexible and robust R2R control in a heterogeneous distributed software environment.

IV. Current Implementation

General Description

A R2R controller has been designed, developed, and tested that meets all of the aforementioned design requirements. At the heart of the controller is a GCC enabling mechanism that provides for a very modular, portable, and adaptable control environment (see Section 2 and [11, 12, 10]). The current implementation operates in a distributed environment that can include a heterogeneous mix of hardware and software platforms as shown in Figure 3. Note that, with this implementation, the controller can integrate any number of software elements in the heterogeneous environment as necessary to achieve R2R control. The integration methodology is described in more detail in Section 5.

GCC: Kernel and Modules							
Mach				Solaris	Win95	WinNT	OS/2
Intel	SPARC	HP	NeXT	SPARC Intel	Intel	Intel Alpha	Intel

Figure 3: GCC Operating Environment

Run-to-Run Control Setup

In order to describe the method in which the GCC enabled R2R controller meets the design goals identified in Section 3, it is first necessary to describe the components of a typical R2R controller. Software elements of a typical R2R controller were introduced earlier in Figure 1. All R2R controllers must include the following five components

(1) A metrology interface component: As R2R control is a form of feedback control, it requires measured output data. The metrology interface component could be automated (e.g., obtaining data from a metrology unit via a SECS interface and communicating it to the R2R controller via a TCP/IP link), manual (e.g., prompting a user to type in metrology data obtained manually), or any variation between these two extremes.

(2) One or more R2R optimization and / or control algorithms: This component utilizes metrology information and some form of knowledge of the process to make recommendations on how to modify equipment and/or process inputs so as to optimize or control the process. Generally the algorithm(s) utilizes the history of the process in some form. The methods utilized by these algorithms vary widely, from simple SPC alarm reporting to heuristically-based optimal solution searching. As explained in Section 3, more than one algorithm may be required in the controller so that the system may provide optimization / control over a required domain; these algorithms must be utilized in a complementary fashion.

(3) A recipe download component: This component facilitates the communication of recipe advice information from the R2R controller to the equipment controller. This can be accomplished for example through an automated communication network link or via a graphical user interface (GUI) presenting the recipe advice to a user.

(4) A process monitoring component: Once a recipe is downloaded to a process, processing may begin. Metrology may be conducted for this run only after processing has completed. Thus some form of synchronization is required between the process and the R2R controller. This duty is performed by a process monitoring component. In its simplest form, this component could just be a trigger to the R2R controller indicating that the metrology data for the next run is now available (e.g., a key stroke). In a more complex form, the component could monitor the process in-situ, and generate events to the R2R controller as necessary so as to address warnings, alarms, etc., in addition to normal processing.

(5) A central control navigation component: the controller must contain a core component that coordinates the information of the other four components to effect R2R control. In the simplest form, this navigation component could be developed as a software program that provides non-robust and inflexible R2R control. In most applications it is required that this navigation component provide a dynamic and flexible control environment; the GCC enabler provides such an environment and serves as the navigation component in the current implementation.

The GCC-enabled R2R controller provides a system that incorporates the above five elements while additionally allowing other software elements to be incorporated as necessary. The system can be robust in its domain of application to a specific process as it allows for the complementary utilization of multiple optimization and control algorithms (see Figure 1 and [1]). It is also robust with respect to its methods of R2R control because the GCC database is capable of storing complex discrete control scenarios. The system is distributed, hardware/software platform independent, and portable as it can exist over a heterogeneous network of computer types (see Figure 3). Further, the system is process-independent as it is able to isolate (through partitioning of data in the GCC database) the portion of R2R control that is process specific from the portion that is process generic, thus maximizing re-usability of control software and knowledge.

Other qualities of the controller include a capability of incorporating software elements, or "modules", as necessary from a variety of sources including (third party) commercial; the mechanism through which these elements are incorporated into the GCC is straightforward and is described further in Section V. Also, the system can be constructed from off-the-shelf components and can thus be set up and maintained for an industrial application.

Implementation Details

The qualities of the GCC-enabled R2R controller are further illustrated through a detailed discussion of the software system developed and its operation. A block diagram of the system is shown in Figure 4. This figure shows that a number of software elements, or "modules", have been incorporated to form a R2R control system. Metrology information is input to the system through a Metrology module. The control algorithm utilized is a gradual mode linear approximation algorithm developed at MIT [3, 4, 17]. An Equipment Module serves as the interface to the equipment controller and is the recipe download component. For this implementation, a human determines when the process was complete and metrology should be performed, thus the process monitoring component is a GUI to the user.

Other aspects of the implementation depicted in Figure 4 include a model builder module which provides a mechanism for setting up the R2R control problem formulation [4]. A simulator module is provided for "what if" analysis and to verify that the process and controller are performing acceptably. A fuzzy logic mechanism associated with the database allows for the support of fuzzy control. This type of control is necessary when there is insufficient information available to make a "crisp" decision and consequently a "best" course of action must be determined (note that the availability of fuzzy logic for control does not in any way preclude the utilization, exclusively or non-exclusively, of crisp (deterministic) control). As an example, if multiple algorithms are available for control and the domains of applicability of these algorithms are delineated with heuristics, then the decision of which algorithm provides the "best" advice for a particular run may be arrived at through the use of fuzzy logic [18].

As a final note to Figure 4, it can be seen that the GCC depicted is part of a hierarchical network of GCCs. That is, a GCC can control or be controlled by another GCC. Indeed the GCC may be utilized for any form of discrete control and thus a hierarchical network of GCCs could be utilized to provide multiple layers of control in the semiconductor manufacturing facility [11, 19].

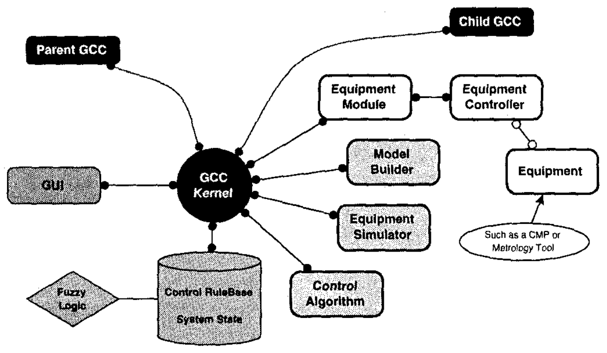


Figure 4: Implementation Block Diagram

Operation

In the current implementation, the various modules and the GCC kernel have associated GUIs that provide an interface to the user. In the following paragraphs, the operation of the GCC-enabled implementation as a R2R controller is described by first presenting some of these GUIs, and then describing the use of these GUIs in performing one "loop" of R2R control.

GCC Environment: The GCC-enabled R2R controller is implemented as a suite of applications, which could include third-party applications. These software applications consist of a GCC kernel and a set of *modules*. Each of the modules interact with the GCC kernel via a well-defined object-oriented interface (see Section 5). As an example, a typical GCC implementation is shown in Figure 5. Each of the applications may be run by double-clicking on its icon.

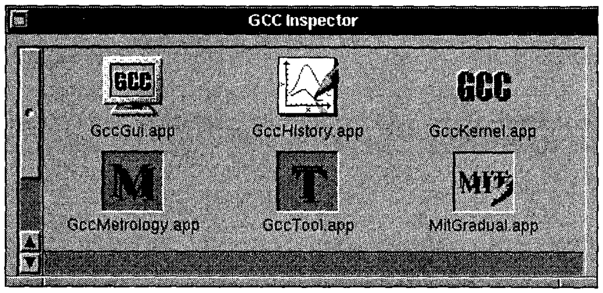


Figure 5: Typical GCC Implementation

GCC Tool GUI: If the user double-clicks on a particular equipment module, an associated equipment connection window appears. This window, shown in Figure 6, displays (1), the software modules that are utilized in the control of that particular equipment instance, and (2), the control commands that may be initiated by the system user.

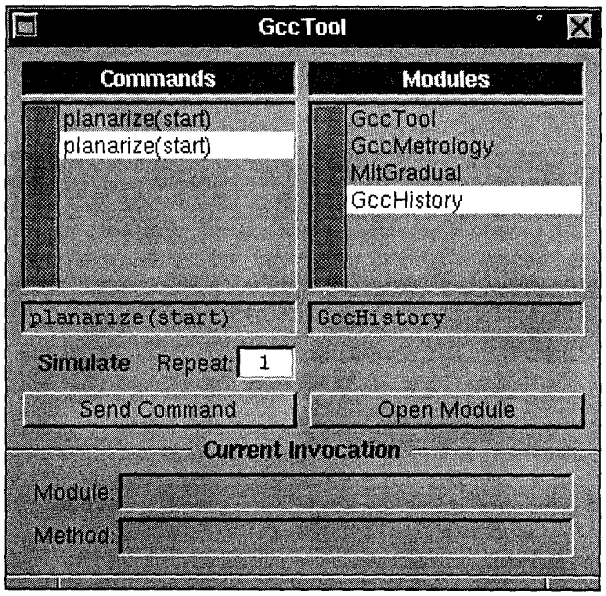
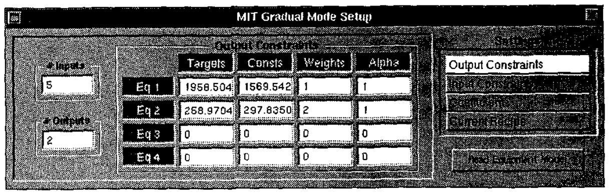


Figure 6: GCC Tool GUI

MIT Gradual Mode GUI: From a GCC Tool window, a user may open GUIs that are available for any of the modules. Note that these GUIs are generally supplied by the third party that developed the module, but are incorporated into the GCC system utilizing an interface specification (see Section 5). As an example, a portion of the interface to an MIT Gradual Mode Module is depicted in Figure 7. Note that this nested interface can display the entire R2R control model and provides GUIs for the setup of process models and formulation of process simulations.


MIT Gradual Mode Setup

Output Constraints

Inputs: 5

Outputs: 2

	Targets	Consts	Weights	Alpha
Eq 1	1958.504	1569.542	1	1
Eq 2	258.9704	297.8350	2	1
Eq 3	0	0	0	0
Eq 4	0	0	0	0

Settings: Output Constraints

Figure 7: Gradual Mode Module GUI

GCC Operation: Once the hardware and software environment has been set up for the GCC-enabled controller, R2R control proceeds as follows. The GCC and all appropriate modules are "started" by double clicking on their respective icons (see Figure 5). Before the first control run, the R2R problem formulation is set up (initialized) through the appropriate GCC Tool window (Figure 6) by opening the control module GUI (e.g., MIT Gradual Mode) and providing the appropriate model and historical information. (Note that the control module may be running on a different computer or even a different operating system (see Figure 3), but its GUI will automatically be displayed on the operator's screen.) After the model is set up, an initial recipe is downloaded to the tool and the system is ready to perform R2R control. This is accomplished as follows. A command is sent to the tool via the GCC Tool window to process a wafer (e.g., planarize). The process is monitored for completion, whereupon the GCC system is notified either automatically by the equipment controller, or manually by a user monitoring the process. The GCC then automatically presents the user with a metrology interface for the entry of metrology data. This data is taken by the GCC and sent to the controller algorithm, which returns a recipe advice. The GCC then downloads this new advice to the equipment for the next run, and the R2R control "loop" is completed.

This description depicts a very straightforward implementation of R2R control, however its serves to illustrate GCC R2R implementation concepts such as process-independence, portability and reusability of software, and the capability to incorporate third-party software. There are however a number of GCC system features that warrant further discussion. These features are addressed in the following section.

V. Features

Distributed Dynamic Architecture

The GCC can inter-operate with software programs such as process model builders, optimizers, control algorithms, and equipment controllers. These types of programs are called modules of the GCC. The GCC defines a generic module interface which allows modules dynamically to connect and disconnect from the GCC without any code modification. This module interface facilitates the passing of arbitrary data, determined at runtime, between the GCC and the module. It also allows users to develop custom modules or third-party developers to produce shrink-wrapped modules. This generic and dynamic interface to software modules contributes to the GCC's quality of process independence.

The GCC and all of its components run on multiple hardware architectures and operating systems: Intel 486 and Pentium, HP PA-RISC, and Sun SPARC, Solaris (SPARC, Intel), Windows 95 and NT (Intel), and in the future OS/2 (Intel). Because the GCC is portable across these platforms, the GCC kernel can be running on a Pentium-based computer, for example, while different modules run on several other platforms, yet always displaying their graphical user interface on the terminal where the operator sits-independent of platform.

Integration Interface

The GCC module interface relies on a distributed objects architecture which allows objects to send messages to other objects in other tasks or have messages executed in other threads of the same task [20]. In general, an object sends a message to a remote object by communicating in its own address space with a proxy for the remote object. The proxy assumes the identity of the remote object; it has no identity of its own. The application is able to regard the proxy as if it were the remote object; for most purposes, it is the remote object. Figure 8 depicts an example of remote messaging, where object A communicates with object B through a proxy, and messages for B wait in a queue until B is ready to respond to them.

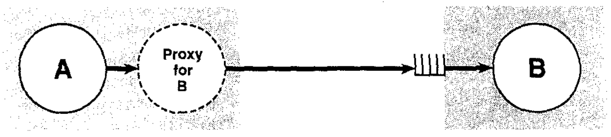


Figure 8. Remote Messages

Note that proxy does not require access to the remote object's class. It isn't a copy of the object but a lightweight substitute for it, transparently passing the messages it receives on to the remote receiver and managing the interprocess communication. Its main function is to make a remote object appear as if it were local. A remote receiver is typically anonymous. Its class is hidden inside the remote application. The sending application doesn't need to know how that application is designed or what classes it uses. It doesn't need to use the same classes itself. All it needs to know is to what messages the remote object responds.

Adaptable Control Scheme

One quality that sequential controllers must possess in the semiconductor manufacturing environment is the ability to adapt to multiple and varying control schemes. Many semiconductor processes are not well understood and process response surfaces are constantly shifting, drifting, and changing shape. The control schemes for these processes likewise must vary with time. Further, many processes and their process controllers are expected to exist in a flexible manufacturing environment. Thus controllers must be able to adapt to their changing environment by navigating through a robust and necessary complex control paradigm during operation. This navigation must include obtaining, during runtime, information from outside sources where necessary so as to adapt to new and unforeseen control situations. As an example, the controller should be able to query an expert user or a neural network as necessary during runtime to formulate responses to unforeseen control events and "learn" how to service these events in the future.

The R2R controller that has been developed provides this capability by utilizing the learning mechanism component of the GCC enabler. This mechanism operates as follows (see also [11, 12]). When an event is received at the controller, the GCC enabler interacts with its database to determine if a control action (i.e., event service routine sequence) has been formulated for the event. If not, i.e., if the knowledge has not been entered into the controller, the controller invokes an interface with an expert user and reports the situation. If the expert user wishes to formulate a control action and "teach" the controller, the GCC displays the list of available control action sequences (that have been formulated for other known events). If the expert user indicates to the controller that one of these action sequences is appropriate to service the current event, the controller invokes that action sequence, and stores the relationship between the event and action sequence and thus "learns" how to service the event.

Otherwise the controller must continue to interact with the expert user to formulate a new action sequence. It does this by displaying the list of modules available to the controller and guiding the user in constructing a sequence of module invocations (including module parameters; see Figure 6). This sequence then becomes the new "learned" action sequence for servicing that event.

VI. Application to R2R Control of CMP

Experimental Setup

The R2R controller developed has been applied to the control of both plasma etch and, more recently, CMP processing. In the CMP system the controller was set up to provide multi-variate R2R control of the within wafer uniformity and removal rate of unpatterned wafers, by suggesting values for input CMP parameters of speed, force, pressure and profile. The 4 input by 2 output process model that is utilized for control was derived empirically utilizing design of experiments and regression analysis. The modules utilized in the controller are listed in Figure 6. Specifically, the control algorithm utilized is a gradual mode linear approximation controller developed at MIT [4]. The Metrologer module provides a GUI for manual entry of metrology data (removal rate and non-uniformity). The Recipe Downloader interacts with the tool controller to provide recipe control advice. The other modules listed in Figure 6 maintain synchronization between the tool and R2R controllers, and update the R2R controller database and user as to the history of the process.

Results

The R2R controller has been applied to the control of a number of CMP processes. As an example, Figure 9 shows the results of a typical experiment involving 45 wafers in a controller CMP process, along with empirical estimates of open loop (uncontrolled) operation derived from earlier experiments. As the figure illustrates, the controller accurately compensated for removal rate drift while maintaining an acceptable level of uniformity.

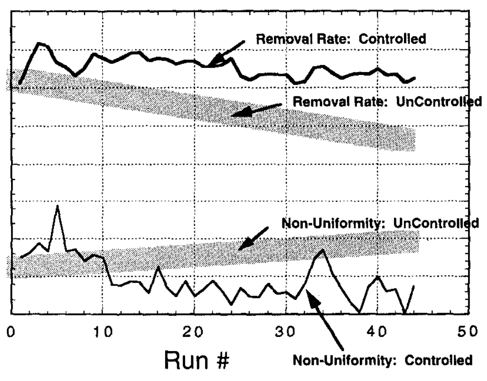


Figure 9: Experimental Results in R2R Removal Rate and Uniformity Control in a CMP Process

VII. Conclusions and Future Work

A process-independent run-to-run controller has been developed and successfully applied to the control of a chemical-mechanical planarization process. In addition to being process independent, the R2R control implementation is software / hardware platform independent, distributed, portable, robust with respect to support of complex control sequences as well as coverage of the process space, supports the incorporation of third party software elements, and may be developed utilizing off-the-shelf technology. These qualities are all considered design requirements of a R2R controller operating in a semiconductor manufacturing environment. The implementation meets these requirements due mainly to its utilization of a Generic Cell Controller core.

The implementation has been demonstrated to provide improved multivariate (removal rate and uniformity) control of a number of CMP processes. This represents a major step in the development and testing of the GCC enabled controller, however numerous areas of future research remain. The capability of the controller to utilize a multitude of optimization and control algorithms will be pursued further; this effort will not only demonstrate the robustness of the controller, but will also help to identify the control domains of the respective algorithms and the combinations of algorithms that are suitable for robust control of various processes. Another major focus of future efforts will be the incorporation of the R2R controller into an automated hierarchical factory control scheme. Efforts are currently underway to explore, develop and deploy low-level in-situ (real-time) control techniques for selected processes, as well as high level inter-cell control for the factory. The long term goal of these efforts is the development of a multi-level control framework for the factory, and a subsequent implementation that concurrently utilizes inter-cell, R2R and real-time control.

Acknowledgments

The authors gratefully acknowledge the technical assistance of Nauman Chaudhry for setting up and maintaining the current GCC database; and Duane Boning, William Moyne and Taber Smith for the developing the current version of the MIT gradual mode algorithm. This research has been supported in part by SEMATECH and by the Center for Display Technology and Manufacturing.

References

[1] J. Moyne, H. Etemad, and M. Elta, "A Run-to-Run Control Framework for VLSI Manufacturing," Proc. Microelec. Processes, Sensors, and Controls, SPIE, vol. 2091, 1994, pp. 379-390.

[2] B. Rashap, M. Elta, H. Etemad, J. Fournier, J. Freudenberg, M. Giles, J. Grizzle, P. Kabamba, P. Khargonekar, S. Lafortune, J. Moyne, D. Teneketzis and F. Terry, Jr., "Control of Semiconductor Manufacturing Equipment: Real-Time Feedback Control of a Reactive Ion Etcher," *IEEE Transactions on Semiconductor Manufacturing*, August 1995.

[3] D. Boning, W. Moyne, T. Smith, J. Moyne, A. Hurwitz, "Practical Issues in Run by Run Process Control," to be presented, Sixth Annual SEMI/IEEE ASMC, Boston, November 1995.

[4] W. Moyne, "Run by Run Control: Interfaces, Implementation, and Integration," S. M. Thesis, MIT EECS, May 1995.

[5] S. W. Butler and J. Stefani, "Application of Predictor Corrector Control to Polysilicon Gate Etching," Proc. Amer. Control Conf., June 1993.

[6] C.W. Moreno, "Self-Learning Optimizing Control Software," *ISA Proceedings*, June 1986.

[7] E. Del Castillo and A. Hurwitz, "Run to Run Process Control: a Review and Some Extensions," submitted to J. Qual. Tech., February 1994.

[8] A. Hu, X. Zhang, E. Sachs, and P. Renteln, "Application of Run by Run Controller to the Chemical-Mechanical Planarization Process, Part I," IEEE Proc. of the 15th Int. Elect. Manuf. Tech. Symp., October 1993.

[9] A. Hu, H. Du, S. Wong, P. Renteln, and E. Sachs, "Application of Run by Run Controller to the Chemical-Mechanical Planarization Process, Part II," IEEE Proc. of the 16th Int. Elect. Manuf. Tech. Symp., October 1994.

[10] R. Telfeyan, J. Moyne, A. Hurwitz and J. Taylor, "Demonstration of a Process-Independent Run-to-Run Controller," Electrochem. Soc. Meeting, June 1995.

[11] J. Moyne and L. McAfee, Jr., "A Generic Cell Controller for the Automated VLSI Manufacturing Facility," IEEE Trans. Semi. Manuf., vol. 5, no. 2, May 1992, pp. 77-87.

[12] J. Moyne, "Generic Cell Controlling Method and Apparatus for Computer Integrated Manufacturing System," *Patent Pending*, filed with the United States Patent and Trademark Office, (Filed, August 1991; Formal notification of allowance, May 1995).

[13] D. Boning, A. Hurwitz, J. Moyne, W. Moyne, S. Shellman, T. Smith, J. Taylor, and R. Telfeyan, "Run by Run Control of Chemical Mechanical Polishing," (to be presented at) International Electronics Manufacturing Technology Symposium, Austin, TX, September 1995.

[14] R. Jairath, J. Farkas, C. Huang, M. Stell, S. Tzeng, "Chemical-Mechanical Polishing: Process Manufacturability," *Solid State Technology*, July 1994, pp. 71-75.

[15] P. Singer, "Chemical-Mechanical Polishing: A New Focus on Consumables," *Semiconductor International*, February 1994, pp. 48-52.

[16] M. Martinez, "Chemical-Mechanical Polishing: Route to Global Planarization," *Solid State Technology*, May 1994, pp. 26-31.

[17] E. Sachs, A. Hu, and A. Ingolfsson, "Run by Run Process Control: Combining SPC and Feedback Control," IEEE Trans. Semi. Manuf., vol. 8, no. 1, February 1995, pp. 26-43.

[18] J. Moyne, N. Chaudhry, R. Telfeyan, "Adaptive Extensions to a Multi-Branch Run-to-Run controller for Plasma Etching," *Journal of Vacuum Science and Technology A,* Vol. 13, No. 3, May/June 1995, pp. 1787- 1791.

[19] N. Chaudhry, J. Moyne, and E. Rundensteiner, "A Generic Framework for Inter-Cell Control of a Semiconductor Manufacturing Facility," (to be presented at) *42nd National Symposium of the American Vacuum Society*, Minneapolis, MN, October 1995.

[20] NeXT Computer, Inc. *NeXTSTEP Object-Oriented Programming and the Objective C Language, Release 3.* Reading, Mass: Addison-Wesley, 1993.