Recipe Management: A Matter of Efficiency

Todd C. Williams, *Compaq Computer Corporation, Camas, WA, USA*

Abstract

This article is a discussion of the needs of a recipe manager in today's fabs. It enumerates the high-level requirements for a full-featured recipe management system. It also describes the interrelationship of the functions, methods of utilization, and the incorporation of a recipe manager in the overall manufacturing system.

Introduction

Up to now, recipe management systems have consisted of the configuration control and upload/download aspects of recipe management. These functions, although needed, are woefully inadequate for what has been requested by fabrication facilities. Additional functions that consider the equipment condition, equipment capabilities, production line loading, scheduling, and processing time are a few of the items that are required. Furthermore the tight link between the operation and a single, unique recipe ID is no longer adequate for the complex equipment on the production floor. Recipes are more aptly described by the function the recipe performs for a given process.

Why Focus on Recipe Management?

Recipe management is a rarely discussed function. Why? To begin with, a full-featured recipe manager is difficult to provide without full factory integration. As fabs became more integrated, the major function that was required became recipe upload/download. This required recipe storage. Recipe storage allowed for rudimentary configuration control and sign-off. Still, today, a lot of equipment does not allow recipe upload/download, recipe standards do not fully exist, those that exist are not adhered to, and therefore the full benefits of a recipe manager cannot be realized.

Back to the question: recipes, like other resources throughout the fab, are required prior to making a process run. Without the right recipe, the fab loses efficiency via lost time, rework, or scrapped product. Increasing efficiencies and throughput is critical in reducing the cost of the product. To do this, information about the recipe is required.

The Right Recipe at the Right Place

Ensuring that the proper recipe is available for a production run prior to the material being delivered to the equipment is a requirement. Many facilities assume that the recipe is actually on the equipment prior to dispatching the lot. Upon the lot being placed on the input port, it may be determined that the recipe does not exist, has been deactivated, or has been changed, so that the lot cannot be run. Even worse, the lot is run – possibly destroying the wafers.

**Table 1.
Recipe Management Functions**

Function	Enables
Store recipe	• Recipe distribution • Configuration control • Recipe verification
Store checksum	• Recipe verification
Configuration control	• Control of released recipes • Change notification • Modification analysis
Parameter assignment	• Processing adjustment • Range control • History tracking • Parameter control
Name resolution	• Dynamic choice of recipe • Equipment optimization
Recipe verification	• Recipe existence • Recipe integrity
Recipe modification	• Alert for unauthorized change • Automatic configuration control
Recipe characteristics	• Dynamic allocation of recipes • Scheduling based on varied equipment configuration
Recipe attributes	• Controlled dynamic use of recipe • Valid/expiration times

The Travelling-Salesperson Dilemma

There is an old question of how to minimize the cost and maximize the efficiency of a salesperson on a multiple-city trip: the lowest travel cost and the least amount of travel time, giving maximum exposure to the customer. The same problem exists in the fab. Many lots with dissimilar processes have to run through the same equipment. How does one configure the equipment to get the overall best throughput? The matrix of whether a machine is configured for pre-metal or post-metal or split down the middle requires knowledge of not only the material to be processed but also details about recipe attributes and its availability.

What is in a Name?

Routes are tied to a recipe. What if a recipe has to be changed to accommodate the equipment's configuration? The "one recipe for one process" model fails. What really determines the recipe to be used is a set of attributes for the process. These attributes must be matched to the attributes for the recipe. In this way, a dual-chamber recipe is equivalent to a quadruple-chamber recipe as long as the same function is performed. The difference is the speed at which the process is performed. The scheduling system, which has a view of the entire fab, makes the decision on the basis of the overall efficiency of the fab, not the lot.

BASIC FUNCTIONS

The aforementioned issues must be handled by a full-featured recipe management system. For ease of discussion they will be broken into two sections – basic and advanced functions. In these categories, the major requirements will be discussed. A short list of functions is shown in Table 1.

The basic functions for a recipe management system are the functions present in most systems today – recipe body storage, configuration control, and parameter assignment. Let's spend a few minutes discussing the basic functions.

Recipe Body Storage

Being able to store the body of a recipe is the cornerstone of recipe management. With the storage of the recipe body, the recipe manager is the enabler that allows the computer-integrated manufacturing (CIM) system to have control over the recipe. Recipes are available for download or comparison to recipes on equipment. Recipes may be distributed to equipment other than the equipment on which they were created. This distribution may follow well-defined business rules instead of other, error-prone methods.

The types of business rules that may be enforced depend on the equipment, the process, and the methodology of how the equipment will be run. Changes in the recipe body can be caught and actions taken on that error. These actions may include:

- always overwriting the recipe with the "recipe manager" version of the recipe
- overwriting based on equipment type
- initiating an alarm on the equipment for human intervention
- stopping all processing using that recipe until an engineer has corrected the problem.

More importantly, these actions need to be dynamic and configurable on the basis of product, route, equipment, tool ID, or some other basis.

The goal is to eliminate the possibility of a recipe being changed and inadvertently affecting wafers. The recipe manager needs to ensure that all wafers receive the same process time after time.

Offline recipe storage provides other benefits, too. By maintaining proper versioning, engineering has one more tool for determining if a recipe change has caused a positive or negative effect on the wafers. Has a change three versions ago combined with a more recent change caused a cumulative effect on the yield? Access to the data online allows this type of analysis.

Configuration Control

Configuration control is a misunderstood process. Contrary to the popular opinion that configuration control is a bureaucrat's dream, it was really born out of a need to notify everyone of a change and to ensure that the change being instituted was actually consistent with the product direction.

Configuration control needs to notify the appropriate people of a change and ensure that one change being made is consistent with another. Much of this process is based on the way a company implements a configuration control process. Having an engineering vice president sign off a change in oxide thickness may not be as valuable as having an implant engineer approve the change. Improper implementation of a change control system will hinder the process rather than enhance it.

Configuration control needs to be dynamic and allow routeing changes based on the changes made to the recipe to ensure that the proper parties see the modification. The recipe manager should allow for integration with the company's current configuration control system.

Not all changes need configuration control. This, once again, is a site-specific requirement. Parameter value changes do not need a sign-off (in fact, it is not even practical). Deactivation of a recipe in most cases does not need sign-off – production and the engineer need a quick method of removing a recipe from the process. Activating a recipe, and changes to film thickness, implant dose, and ramp time of a recipe need configuration control. The recipe management system needs to allow flexibility in this process. The user needs to be able to quickly and easily select the items that require the configuration control process to be invoked.

The configuration control system must be able to supply an electronic signature. The signature will be associated with each item in the recipe manager that requires approval (on the basis of the user's business rules). For transactions that require that a sign-off be completed, this signature and type of transaction can be passed to the configuration control system to confirm that the action may take place. This action is critical for users that are going to use modules other than the base modules supplied with the recipe manager (i.e. an existing equipment maintenance and management system, recipe body storage system, or parameter override system). By using the signature, custom features can be added without compromising the configuration control process.

Parameter Assignment

Almost all processes in the fab need to be "tweaked". Traditionally, in automated fabs, this is done by the operator at the time the process is started. The operator will enter data on the graphical user interface (GUI) associated with the equipment. The data is then downloaded to the tool as part of the recipe or as separate activation to equipment parameters. Some systems save this data for engineering analysis while others do not. This data needs to be saved.

Changes to parameters usually come from metrology operations and are fed to the process that is about to run. In the case of advanced process control, data may be fed in real time. Operators may enter the data manually or the data may be automatically entered via a feedback/feed-forward mechanism.

This process, often called parameter override, has the following requirements:

- Associate parameters with a process that can be overridden.
- Limit the range of values that can be entered.
- Store a default value for the parameter (optionally allow it to be updated to the last value entered).
- Keep a history of the values entered associated with the lot processed, for engineering analysis.

ADVANCED FUNCTIONS

Beyond the basic functions provided by a recipe manager, newer, more advanced requirements arise when one wishes to fully optimize the line. The recipe manager needs functionality to enable many diverse actions in the fab. Data from the recipe manager needs to be used by schedulers, planners, dispatchers, and equipment maintenance systems to ensure the proper utilization of the fab. To do this, a variety of advanced features are required.

Name Resolution

Most manufacturing execution systems (MESs) require that a single-recipe ID be assigned to a process. Although the method of assigning the recipe to the route varies widely, there is a one-to-one relationship. What happens when a piece of equipment needs a different recipe? What if the equipment is only going to run one chamber for a process instead of two and a different recipe is required to do this? Solutions range from copying one recipe to another keeping the name the same, to custom modules and "intelligent naming" schemes.

The requirement being fulfilled is "name resolution". In name resolution, the type of process being done and the present conditions in the fab are used to determine the proper recipe to use.

Through name resolution the result of the process is described (e.g. 50 angstrom thermal oxide). The recipe manager then returns a list of recipes and conditions that can perform that process (Table 2). Although the best examples are provided by multichambered equipment (see below), other situations may require this. In the case illustrated in Table 2, equipment from different manufacturers can be used to grow the same oxide. The naming rules on these tools are not the same; therefore name resolution is needed to determine the available recipes.

Recipe Attributes

Recipe attributes describe how and when a recipe is to be used. These attributes range from the version number to recipe expiration times.

Each recipe must have a set of attributes that describe it. Although some of these attributes are common from site to site (e.g. version number), other attributes will be site-specific and the recipe manager will require user-defined fields to be added and maintained. These attributes will be used in the recipe selection process and in name resolution. Some examples of attributes are:

- version number
- recipe expiration date
- recipe activation date
- subrecipe ID
- subrecipe ID list
- recipe type (i.e. recipe list or process recipe)

TABLE 2.
EXAMPLE OF RECIPE NAME RESOLUTION

Equipment ID	Recipe ID	Process
FRN-KOK-001	50GOX	50 angstrom thermal oxide
FRN-TEL-001	GateOxide	50 angstrom thermal oxide

TABLE 3.
EXAMPLE OF EQUIPMENT CONFIGURATION OPTIONS

Equipment ID	Configuration option 1	Configuration option 2
CVDMT001	OneChamber	ThreeChamber
CVDMT002	OneChamber	0
CVDMT003	TwoChamber	0

Recipe ID	Equipment option 1	Equipment option 2
DEPAL2W	TwoChamber	ThreeChamber
DEPAL1W	OneChamber	N/A

In addition, various recipe styles must be accommodated. Beside the recipe being human-readable or not, it must be possible to describe recipes as linked or non-linked, recipe lists, or other types of recipes that equipment manufacturers have introduced.

Equipment Capability

One of the biggest uses of name resolution is based on equipment capability. Equipment, especially multichambered equipment, takes on the look and feel of different pieces of equipment throughout the processing day. Chambers can malfunction and go offline, or the equipment may be "wounded" but still capable of processing material and keeping a level of production running until a more convenient time becomes available for maintaining the equipment.

One scenario is that it may benefit production to split the functionality of a tool. Some processes must be isolated. For example, pre- and post-metallization processes cannot be mixed through a piece of equipment. However, by isolating chambers, half of the equipment may be premetal while the other half is post-metal. This procedure gives production more flexibility in scheduling the floor.

An example is in order. Assume there are three identical pieces of equipment that are configured differently. CVDMT001, CVDMT002 and CVDMT003 are the same make and model of equipment. The equipment can only run aluminium on some of the chambers. CVDMT001 can run aluminium on one or three chambers while CVDMT002 can only run on one chamber (Table 3). There are sets of recipes that can run on various equipment configurations. In this example, there are two recipes: DEPAL2W can deposit aluminium using two chambers and DEPAL1W can deposit using only one chamber. Therefore, DEPAL2W can run on CVDMT003 and CVDMT001 since they allow two- and three-chamber processes, respectively.

Recipe Availability

Recipe availability is the next piece of data that is critical. Yes, in an ideal world the recipes are all in the repository and can be downloaded to the equipment. The odds of the perfect world appearing in a semiconductor fab are, however, a little slim. Two primary conditions exist to mess up our world – the equipment cannot support up/download or the recipe on the equipment has been changed.

As we will discuss later, one of the major hindrances is that the equipment interface from the supplier may not support the recipe upload/download function. At this point, the recipe manager may "believe" the recipe is on the equipment, when in reality someone has deleted or renamed the recipe. Therefore, the recipe manager notifies the requesting component of the CIM system that the recipe is on the equipment. Not until the lot is placed on the input port does the reject occur. Valuable production time has been wasted.

To solve this problem the recipe manager must be able to query the tool to determine whether the recipe exists on the equipment. This obviously takes more than simply setting up the recipe manager to handle the request – the cell controller must also be modified.

Furthermore, the implementation in the recipe manager must be done properly. The problem is that this function when improperly implemented can create a severe overload on cell-level controllers. Without care and safeguards put in place, scenarios can be developed where hundreds or thousands of calls are submitted to cell controllers asking for recipe existence, slowing down the entire system. Therefore, safeguards must be placed in the recipe manager to generate an alert or prevent this from happening. The recipe manager must have this option configurable and a process must watch to ensure that "wildcard" requests are stopped.

The second issue is one of company policy. What does one do when a recipe has been changed on a piece of equipment and the recipe manager is not aware of it? The configuration control process has been circumvented. There are three basic options – do nothing, always download the "official" version, and raise an alarm for the machine.

If changes to recipes on equipment are to be monitored, then, there are a variety of ways to meet this requirement. One common way is to store a checksum value in the recipe manager, and upload and recalculate the checksum on the current recipe. A difference generates an error.

The next issue is what to do with the error. This could be as simple as to alert the notification system or could be as drastic as raising an alarm that will deactivate the recipe and disallowing its use on any product. Although the action to be taken is a business rule, the recipe manager should support user-defined mechanisms to allow the disabling of a recipe for a piece of equipment or for any use.

Recipe Modification Alarms

To help circumvent last-minute notifications of changes to a recipe, the recipe manager must be able to react to alarms for equipment whose recipes have changed. SEMI E-42-compliant systems that support recipe change alarm are required. In addition, equipment interfaces must be kept up and running to support the trapping of the alarm. Once this alarm is trapped, it may be handled in the same way as discussed earlier. The advantage is that the reaction time is lengthened, hopefully allowing for a less severe action to take place.

Offline Recipe-Editing Support

With the goal of the recipe manager being to optimize fab utilization, it must support offline recipe editing. Various companies provide offline recipe editing. This frees the floor equipment from being used for the tedious job of writing and initial testing of the recipe. Offline recipe editing is its own subject and will not be covered in detail here.

Integration Capabilities

Integration is a complete subject in its own right. However, certain issues need to be addressed in a recipe manager as well as for any component in the CIM system. An interface requires support for a variety of communication methods. Fortunately, the list of frequently used tools for integration is relatively small for the semiconductor industry. Although standards are ultimately the answer, additional layers of code should be in place to assist in the integration of the system. Any component should support both COM and CORBA as well as other, more "raw" communication methods as TIBCO's Rendezvous, BEA Systems' BMQ (formally DECmessageQ), and Microsoft's MSMQ. Support for interfacing with these methods will reduce the task of maintaining the interface or the work required in a potential bus migration in the future.

In addition, all components should have the ability to map data elements. Mapping data elements is simply a mechanism in the component interface that allows one or more fields from outside the component to be mapped to one or more fields inside the component. Although this function is available in some middleware software (e.g. Compaq's BusinessBus or Crossworlds), the rudimentary functionality should be included in the component interface. Alternatively, bundling one of these third-party products is an acceptable option.

USAGE OF THE SYSTEM

The recipe manager is one more tool in the CIM engineer's tool bag. It is an enabler. Properly designed, with a generic bus interface, it should provide some "out of the box" functionality. However, as with any part of a greater CIM system, it must be integrated. This means that other components, such as the cell controllers, schedulers, and dispatchers, need some level of configuration or customization to use these features.

The recipe manager, as with all CIM components, should have mechanisms to minimize the impact of implementation. Features such as bus-independent interfaces and data element mapping can help make this process much easier.

Schedulers and Dispatchers

As mentioned earlier, the goal of a recipe manager must be to improve efficiency. One way to do this is to supply the schedulers and dispatchers with data on the present capabilities on the fab floor. The type of data that should be supplied is:

- recipe availability with respect to equipment capabilities
- recipe availability with respect to equipment configuration
- recipe run times
- all possible recipes for a process, on the basis of the available equipment and the equipment's configuration.

This type of data allows optimization of the floor by allowing complex algorithms to be used that look at various equipment configurations, the length of time for an equipment configuration changeover, and the time savings to production. From this result, decisions can be made that will allow for optimizations of the fab floor.

Having data like this will allow a system to select the optimal equipment configuration or run sequence when there are three lots of product "A" and one lot of product "B". Each of these lots requires a given piece of equipment. They have multiple recipes depending on the number of chambers available, and their run time depends on the number of chambers used. This then creates a situation where a decision needs to be made, mixed with other line priorities and the changeover time on the equipment. This, in conjunction with hundreds of other lots on the line, requires in-depth knowledge of the line and serious computing resources.

Advanced Process Control

The use of advanced process control (APC) requires tools to adjust and control processes to accommodate wafer-to-wafer fluctuations on the line. Recipes and recipe management systems need to be able to accommodate APC systems to allow for and monitor adjustments required by the process. Presently, most parameter data is input by operators prior to the production run. In future fabs, input directly from the APC system will be required. Controls to ensure the data can be overridden and is within limits will be needed not only in the APC system but also in the recipe management system.

Cell Controllers

Cell controllers, the part of the CIM software that interfaces with the equipment controller, will be the heaviest user of a recipe management system. Current recipes, validation checks, parameters, and up/download all require the recipe management system.

REQUIRED EXTERNAL FUNCTIONALITY

Equipment

Still the biggest issue in the way of a proper implementation of a recipe manager is the functionality provided by the equipment manufacturer in the equipment interface.

The cornerstone of functionality is recipe upload/download functionality. This must be implemented on most controllers on the equipment and on the cell controller. However, other items are also required to implement the features discussed. Alarms from the equipment when recipes are modified, equipment capability is changed, or the configuration is altered are required. Being able to query the equipment for its state, a recipe's state, or what the tool is processing at any point in time must be implemented in all equipment interfaces. These interfaces must be consistent, standard, fully documented, well tested, and of high quality. Without this, the tasks that are discussed in this article are difficult, if not impossible, to perform.

ENABLING OTHER SYSTEMS

The recipe manager is just a small part of the overall CIM system. It enables certain functionality, but cannot stand alone. The most important accompanying system is the equipment maintenance and management system (EMMS).

Equipment Maintenance and Management Systems

Complete usage of a recipe manager requires a tight link with an EMMS. In the absence of the EMMS, basic, rudimentary functions should be provided by the recipe manager. These functions are required to give the general state of the equipment and chambers. The ability to determine the capabilities and configuration of the equipment is required in order to determine what recipes can be used for a process to ensure proper utilization of factory resources. The information as to whether a piece of equipment is offline, wounded, or has multiple configurations, joined with a strong recipe manager functionality, gives the factory systems greater flexibility in ensuring high utilization.

CONCLUSION

To decrease the production costs, fabs must either fit more die on a wafer or decrease their cycle time.

To increase the number of die, one must shrink the product or make the wafer larger. In making the wafer larger, automated handling systems are required to get the wafers to the equipment. Prior to scheduling the material to a piece of equipment, all the resources must be available. The recipe manager is the component that supplies the required data for the recipes on the equipment.

Another method to improve efficiency is to decrease cycle time. This may be done by improving resource availability, reducing rework, and reducing scrap.

Reducing wasted time due to improperly configured equipment or inadequate resources available for processing requires systems that are more sophisticated. For the recipe management system, this requires interfacing with schedulers and dispatchers to supply the data (more data than is generally available today) to these systems. The relationship of the equipment configuration to the recipe is critical in this process.

Recipe validation processes (configuration control, checksum verification, parameter override, etc.) are required to ensure that the proper processing of the wafers is done. Without the functionality described here, these processes would be impossible.

A full-featured recipe manager is one part of a complete CIM system. The functionality must be present in order to meet the next generation of fab automation – the next generation that will be required for 300 mm.

ABOUT THE AUTHOR

Todd C. Williams is a Solution Architect for COMPAQ (previously Digital) and has worked on CIM installations at TSMC, Winbond, WaferTech, MASCA, and Komatsu Silicon America. Presently, Mr Williams is working in the Industry Solutions Division of COMPAQ designing solution sets for the semiconductor industry. His responsibilities include the development of an MES-independent recipe management system.

IF YOU HAVE ANY ENQUIRIES REGARDING THE CONTENT OF THIS ARTICLE, PLEASE CONTACT:

Todd C. Williams
Compaq Computer Corporation
833 NW 24th Avenue
Camas
WA 98607-9365
USA

Tel: +1 (360) 834-7361
Fax: +1 (360) 834-0574
E-mail: todd.c.williams@compaq.com

[Reader Ref. 6]