AGENT-BASED CONTROL ARCHITECTURE

TECHNICAL FIELD

This invention relates generally to the field of semiconductor device manufacturing and, more particularly, to an automated manufacturing environment using autonomous, active, software agents to control recipes for processing wafers.

BACKGROUND ART

There is a constant drive within the semiconductor industry to increase the quality, reliability and throughput of integrated circuit devices, *e.g.*, microprocessors, memory devices, and the like. This drive is fueled by consumer demands for higher quality computers and electronic devices that operate more reliably. These demands have resulted in a continual improvement in the manufacture of semiconductor devices, *e.g.*, transistors, as well as in the manufacture of integrated circuit devices incorporating such transistors. Additionally, reducing the defects in the manufacture of the components of a typical transistor also lowers the overall cost per transistor as well as the cost of integrated circuit devices incorporating such transistors.

Generally, a set of processing steps is performed on a lot of wafers using a variety of processing tools, including photolithography steppers, etch tools, deposition tools, polishing tools, rapid thermal processing tools, implantation tools, *etc*. The technologies underlying semiconductor processing tools have attracted increased attention over the last several years, resulting in substantial refinements. However, despite the advances made in this area, many of the processing tools that are currently commercially available suffer certain deficiencies. In particular, such tools often lack advanced process data monitoring capabilities, such as the ability to provide historical parametric data in a user-friendly format, as well as event logging, real-time graphical display of both current processing parameters and the processing parameters of the entire run, and remote, *i.e.*, local site and worldwide, monitoring. These deficiencies can engender nonoptimal control of critical processing parameters, such as throughput, accuracy, stability and repeatability, processing temperatures, mechanical tool parameters, and the like. This variability manifests itself as within-run disparities, run-to-run disparities and tool-to-tool disparities that can propagate into deviations in product quality and performance, whereas an ideal monitoring and diagnostics system for such tools would provide a means of monitoring this variability, as well as providing means for optimizing control of critical parameters.

One technique for improving the operation of a semiconductor processing line includes using a factory wide control system to automatically control the operation of the various processing tools. The manufacturing tools communicate with a manufacturing framework or a network of processing modules. Each manufacturing tool is generally connected to an equipment interface. The equipment interface is connected to a machine interface which facilitates communications between the manufacturing tool and the manufacturing framework. The machine interface can generally be part of an advanced process control (APC) system. The APC system initiates a control script based upon a manufacturing model, which can be a software program that automatically retrieves the data needed to execute a manufacturing process. Often, semiconductor devices are staged through multiple manufacturing tools for multiple processes, generating data relating to the quality of the processed semiconductor devices.

During the fabrication process, various events may take place that affect the performance of the devices being fabricated. That is, variations in the fabrication process steps result in device performance

variations. Factors, such as feature critical dimensions, doping levels, contact resistance, particle contamination, *etc.*, all may potentially affect the end performance of the device. Various tools in the processing line are controlled in accordance with performance models to reduce processing variation. Commonly controlled tools include photolithography steppers, polishing tools, etching tools, and deposition tools. Pre-processing and/or post-processing metrology data is supplied to process controllers for the tools. Operating recipe parameters, such as processing time, are calculated by the process controllers based on the performance model and the metrology information to attempt to achieve post-processing results as close to a target value as possible. Reducing variation in this manner leads to increased throughput, reduced cost, higher device performance, *etc.*, all of which equate to increased profitability.

Configuration control and efficiency issues are prevalent in a distributed computing environment, such as a factory-wide APC system. Typically, there are numerous software developers writing control code to construct the process controllers. One particular developer may work extensively developing controllers of a certain type. It is common for each developer to have a unique programming style, and to rely on routines that they have created themselves. For example, each developer may have a set of routines for interfacing with databases or other entities within the APC framework and for performing various math functions and basic utility functions.

One problem associated with such an arrangement is that there is little consistency between process control scripts. The large number of custom scripts also presents a configuration control problem and an efficiency problem. Developers may spend considerable time duplicating code that has already been developed, perhaps for a different type of process controller that a different developer has created. Debugging non-standardized code is also more time-consuming and further reduces efficiency.

The present invention is directed to overcoming, or at least reducing the effects of, one or more of the problems set forth above.

DISCLOSURE OF INVENTION

One aspect of the present invention is seen in a manufacturing system including a process tool, a controller agent, and a first processing agent. The process tool is configured to process a workpiece in accordance with an operating recipe. The controller agent is configured to determine a control action associated with the processing of the workpiece in the process tool. The first processing agent is associated with at least one of the process tool and the workpiece and configured to interface with the controller agent, receive the control action, and configure the operating recipe based on the control action.

Another aspect of the present invention is seen in a method for controlling a process tool configured to process a workpiece in accordance with an operating recipe. A first processing agent associated with at least one of the process tool and the workpiece is instantiated. A controller agent configured to determine a control action associated with the processing of the workpiece in the process tool is instantiated. The first processing agent is interfaced with the controller agent to receive the control action. The first processing agent is interfaced with the process tool to configure the operating recipe based on the control action.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may be understood by reference to the following description taken in conjunction with the accompanying drawings, in which like reference numerals identify like elements, and in which:

Figure 1 conceptually illustrates a portion of one particular embodiment of a first process flow constructed and operated in accordance with the present invention;

Figure 2 conceptually illustrates, in a partial block diagram, selected portions of the hardware and software architectures, respectively, of the computing devices in Figure 1; and

Figure 3 conceptually illustrates in a partial block diagram the specialization of agents into scheduling agents, processing agents, and controller agents in a second process flow constructed and operated in accordance with the present invention.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and are herein described in detail. It should be understood, however, that the description herein of specific embodiments is not intended to limit the invention to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

MODE(S) FOR CARRYING OUT THE INVENTION

Illustrative embodiments of the invention are described below. In the interest of clarity, not all features of an actual implementation are described in this specification. It will of course be appreciated that in the development of any such actual embodiment, numerous implementation-specific decisions must be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, which will vary from one implementation to another. Moreover, it will be appreciated that such a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking for those of ordinary skill in the art having the benefit of this disclosure.

Referring first to Figure 1, a conceptual illustration of a portion of a process flow 100 constructed and operated in accordance with one particular embodiment of the present invention is provided. The illustrated portion of the process flow 100 includes two stations 105, each station 105 including a computing device 110 communicating with a process tool 115. The stations 105 communicate with one another over communications links 120. In the illustrated embodiment, the computing devices 110 and the communications links 120 comprise a portion of a larger computing system, *e.g.*, a network 125. The process tools 115 are shown in Figure 1 processing lots 130 of wafers 135 in the fabrication of integrated circuit devices. Although the invention is illustrated as it may be used in fabricating semiconductor devices, its application is not so limited, as it may be applied to other types of manufacturing processes. Thus, in the process flow 100 discussed above, the lots 130 of wafers 135 may more generically referred to as "workpieces." The process tools 115 and any process operation performed thereon need not necessarily be related to the manufacture of semiconductor devices in all embodiments. However, for the sake of clarity and to further an understanding of the invention, the terminology pertaining to semiconductor fabrication shall be retained in disclosing the invention in the context of the illustrated embodiments.

Figure 2 depicts selected portions of the hardware and software architectures, respectively, of the computing devices 110 programmed and operated in accordance with the present invention. Some aspects of the hardware and software architecture (*e.g.*, the individual cards, the basic input/output system ("BIOS"), input/output drivers, *etc.*) are not shown. These aspects are omitted for the sake of clarity, and so as not to obscure the present invention. As will be appreciated by those of ordinary skill in the art having the benefit of

this disclosure, however, the software and hardware architectures of the computing devices 110 will include many such routine features.

In the illustrated embodiment, the computing device 110 is a workstation, employing a UNIX-based operating system, but the invention is not so limited. The computing device 110 may be implemented in virtually any type of electronic computing device such as a laptop computer, a desktop computer, a mini-computer, a mainframe computer, or a supercomputer. The computing device 110 may even be, in some alternative embodiments, a processor or controller embedded in the process tool 115. The invention also is not limited to UNIX-based operating systems. Alternative operating systems (*e.g.*, Windows™-based or disk operating system ("DOS") -based) may also be employed. The invention is not limited by the particular implementation of the computing device 110.

The computing device 110 also includes a processor 205 communicating with some storage 210 over a bus system 215. The storage 210 will typically include at least a hard disk (not shown) and random access memory ("RAM") (not shown). The computing device 110 may also, in some embodiments, include removable storage such as an optical disk (not shown), or a floppy electromagnetic disk (not shown), or some other form such as a magnetic tape (not shown) or a zip disk (not shown). The processor 205 may be any suitable processor known to the art. For instance, the processor may be a general purpose microprocessor or a digital signal processor ("DSP"). In the illustrated embodiment, the processor 205 is an Athlon™ 64-bit processor commercially available from Advanced Micro Devices, Inc. ("AMD"), but the invention is not so limited. The 64-bit UltraSPARC™ or the 32-bit microSPARC™ from Sun Microsystems, any of the Itanium™ or Pentium™-class processors from Intel Corporation, and the Alpha™ processor from Compaq Computer Corporation might alternatively be employed. The computing device 110 includes a monitor 240, keyboard 245, and a mouse 250, which together, along with their associated user interface software 255 comprise a user interface 260. The user interface 260 in the illustrated embodiment is a graphical user interface ("GUI"), although this is not necessary to the practice of the invention.

Figure 2 also illustrates selected portions of the software architecture of the computing devices 110. Each computing device 110 includes, in the illustrated embodiment, a software agent 265 residing thereon in the storage 210. Note that the software agents 265 may reside in the process flow 100 in places other than the computing devices 110. The *situs* of the software agent 265 is not material to the practice of the invention. Note also that, since the *situs* of the software agents 265 is not material, some computing devices 110 may have multiple software agents 265 residing thereon while other computing devices 110 may not have any. An automated manufacturing execution system (MES) 270, such as WORKSTREAM™ offered by Consilium, Inc. of Mountain View, CA, resides on at least one computing device 110.

Returning briefly to Figure 1, as was mentioned above, the computing devices 110 may also be part of a larger computing system 125 by a connection over the communications links 120. Exemplary computing systems in such an implementation would include local area networks ("LANs"), wide area networks ("WANs"), system area networks ("SANs"), intranets, or even the Internet. The computing system 125 employs a networked client/server architecture, but alternative embodiments may employ a peer-to-peer architecture. Thus, in some alternative embodiments, the computing devices 110 may communicate directly with one another. The communications links 120 may be wireless, coaxial cable, optical fiber, or twisted wire pair links. The computing system 125, in embodiments employing one, and the communications links 120 are

implementation specific and may be provided in any suitable manner known to the art. The computing system 125 may employ any suitable communications protocol known to the art, *e.g.*, Transmission Control Protocol/Internet Protocol ("TCP/IP").

Referring now to Figure 1 and Figure 2, the software agents 265, collectively, are responsible for efficiently shepherding the lots 130 of wafers 135 through the fabrication process. Each process tool 115 represents some resource that may be employed for this purpose. For instance, the process tool 115 may be a fabrication tool used to fabricate some portion of the wafers 135, *i.e.*, layer, pattern, dope, or heat treat the wafers 135. Or, the process tool 115 may be a metrology tool used to evaluate the performance of various parts of the process flow 100. Thus, the software agents 265 are capable of assessing a plurality of resources for subsequent processing of the lots 130 of wafers 135, allocating the resources represented by the process tools 115, and negotiating among themselves for the allocation of those resources for subsequent processing of the lot 130 of wafers 135.

In the illustrated embodiment, the software agents 265 are self-configuring on start-up, intelligent, state-aware, and imbued with specific goals for which they autonomously initiate behaviors to achieve. The software agents 265 are also self-adjusting as their environment changes. The software agents 265 are implemented as objects in an object oriented programming ("OOP") environment, but the invention may be implemented using techniques that are not object oriented. Their behavior is relatively simple and is script or rules-based. The behavior is designed to achieve selected goals such as achieving an assigned lot due date, achieving a predefined level of quality, maximizing machine utilization, and scheduling opportunistic preventive maintenance. In furtherance of these objectives, the software agents 265 interface with the MES 270 and are integrated with the existing factory control systems (not shown). As will be apparent to those skilled in the art having the benefit of this disclosure, the manner in which this interface and integration occurs will be implementation specific, depending upon the identity of the MES 270 and the factory control systems.

As will be explained further below, the software agents 265 can be specialized on several different levels. One level is by "type," *i.e.*, whether the software agents 265 represent a "consumer" or a "provider" in the process flow 100. More particularly, whether the software agents 265 represent a consumer or a provider is determined by the type of entity it represents in the overall process flow 100. For example, a software agent 265 may represent a lot 130 of wafers 135 (*i.e.*, a "lot agent"), a process tool 115 (*i.e.*, a "machine agent"), or a process material (*i.e.*, a "resource agent"). The software agents 265 may also specialized by function—*i.e.*, by what function the software agent 265 performs in the process flow. For example, a software agent 265 may be configured to perform a particular function without necessarily representing a particular entity. For example, the software agent 265 may be configured to determine control actions (*i.e.*, a "controller agent") for a particular lot 130 of wafers 135 processed in a process tool 115. The actions taken by the controller agent may depend on both the characteristics of the lot 130 and the characteristics of the process tool 115. A controller agent may be invoked by either a lot agent or a machine agent, depending on the particular implementation. In the illustrated embodiment, the controller agent remains associated with a particular process tool 115 throughout its life cycle, however, other arrangements may also be used. Other functions that may be performed by software agents 265 include scheduling.

In one embodiment discussed more fully below, for instance, there are scheduling agents, processing agents, and controller agents. Note that the software agents 265 need not necessarily exist in a one-to-one

correspondence with manufacturing domain entities, such as lots 130, process tools 115, *etc.* Instead, most domain entities are each represented by a group of software agents 265. For instance, a lot 130 or a process tool 115 may have both a scheduling agent and a processing agent. A controller agent may be associated with a particular process tool 115 and may be invoked by one of the processing agents (*i.e.*, associated with the lot 130 or the process tool 115). This arrangement facilitates the design of specialized objects that exhibit specialized behavior to support a single aspect of domain entity functionality.

In this particular embodiment, the software agents 265 are implemented using object-oriented programming techniques. In the terminology of object-oriented computing, a software "agent" is an autonomous, active object. Given its set of operations, a software agent 265 can take independent action in response to local conditions, thereby generating adaptable system behavior. The present invention presents an agent-enhanced system that defines, configures, and deploys autonomous and mobile "software agents" that mimic and improve the functioning of "real world" agents in a semiconductor manufacturing plant such as factory workers, material, equipment, processes, *etc.* One skilled in the art will recognize that an agent or other software object can include one or more software objects. As used herein, the term "object" will be understood to be a software object that may, in turn, be composed of other software objects. Conversely, one skilled in the art will also recognize that the functionality of one object may be combined with other functionalities. It is to be understood that functionalities described as being associated with separate objects may be combined into the functionality associated with a single object.

Furthermore, in this particular embodiment, the software agents 265 are "configurable." In this particular embodiment, the software agents are configurable either by modifying scripts governing their behavior or modifying "control knobs." Control knobs are externally configurable parameters or properties of the software agents that provide flexibility and/or system tuning. For instance, some properties specify curves that guide system decisions. More particularly, the processing and scheduling actions of the software agents 265 are scripted such that certain, predefined events cause execution of a script segment. In one particular implementation, the scripts are implemented in JPython, an open source Java programming language providing a bi-directional Java interface. Additional information, including a download of JPython, are available over the Internet at <http://www.jpython.org/>.

Some portions of the detailed descriptions herein are consequently presented in terms of a software implemented process involving symbolic representations of operations on data bits within a memory in a computing system or a computing device. These descriptions and representations are the means used by those in the art to most effectively convey the substance of their work to others skilled in the art. The process and operation require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantifies. Unless specifically stated or otherwise as may be apparent, throughout the present disclosure, these descriptions refer to the action and processes of an electronic device, that manipulates and transforms data represented as physical (electronic,

magnetic, or optical) quantities within some electronic device's storage into other data similarly represented as physical quantities within the storage, or in transmission or display devices. Exemplary of the terms denoting such a description are, without limitation, the terms "processing," "computing," "calculating," "determining," "displaying," and the like.

Note also that the software implemented aspects of the invention are typically encoded on some form of program storage medium or implemented over some type of transmission medium. The program storage medium may be magnetic (*e.g.*, a floppy disk or a hard drive) or optical (*e.g.*, a compact disk read only memory, or "CD ROM"), and may be read only or random access. Similarly, the transmission medium may be twisted wire pairs, coaxial cable, optical fiber, or some other suitable transmission medium known to the art. The invention is not limited by these aspects of any given implementation.

The process flow 100 may be implemented under an Advanced Process Control (APC) framework. An APC system comprises a distributed software system of interchangeable, standardized software components permitting run-to-run control and fault detection/classification in the process flow 300. The present invention of software agents 265 and their operation is layered on top of the APC System to achieve a nearly autonomously operating semiconductor fabrication process flow.

The software components of the APC System implement an architectural standard based on the Semiconductor Equipment and Materials International ("SEMI") Computer Integrated Manufacturing ("CIM") Framework compliant system technologies and the Advanced Process Control ("APC") Framework. The CIM (SEMI E81-0699 - Provisional Specification for CIM Framework Domain Architecture) and APC (SEMI E93-0999 - Provisional Specification for CIM Framework Advanced Process Control Component) specifications are publicly available from SEMI. SEMI may be contacted at 8310 Capital of Texas Highway North, Suite 290 Austin, TX 78731, phone: 512-349-2422, fax: 512-349-2442, internet address: <http://www.semi.org>.

This particular architecture relies heavily on software utilizing object oriented programming and employs the Object Management Group's ("OMG") Common Object Request Broker Architecture ("CORBA") and CORBA_Services specifications for distributed object systems. Information and specifications for the OMG CORBA architecture are also readily, publicly available.

An exemplary software system capable of being adapted to perform the functions of the APC System as described herein is the Catalyst system commercially offered by KLA-Tencor, Inc. KLA-Tencor may be contacted at: 160 Rio Robles, San Jose, CA 95134, phone: 408-875-6000, fax: 408-875-3030, <http://www.kla-tencor.com>. Additional information regarding the Catalyst system is also available from KLA-Tencor on their website at <http://www.kla-tencor.com/controlsolutions/catalystapc.html>.

Referring now to Figure 3, a simplified block diagram of an exemplary embodiment of a process flow 300 implemented using software agents 265 in accordance with another embodiment of the present invention is provided. An exemplary lot 130 is represented by a lot scheduling agent 310 and a lot processing agent 320. An exemplary process tool 115 is represented by a machine scheduling agent 330 and a machine processing agent 340. The lot scheduling agent 310 negotiates with various entities in the process flow 300, such as the machine scheduling agent 330 to secure the resources needed for completion of the lot. For example, lot scheduling agent 310 and the machine scheduling agent 330 may use a "contract net negotiation protocol" approach to schedule an appointment for the lot 130 with the process tool 115.

The lot processing agent 320 and/or machine processing agent 340 may be invoked when the lot 130 is received at the process tool 115 for processing. The lot processing agent 320 and machine processing agent 340 interface with a controller agent 350 to determine necessary control actions for configuring the operating recipe of the process tool 115 to process the lot 130. Either lot processing agent 320 or the machine processing agent 340 may invoke the controller agent 350. In one embodiment, the lot processing agent 320 may send the controller agent 350 data regarding the characteristics of the lot 130 (*e.g.*, dimensional metrology data, electrical metrology data, *etc.*) that is needed to determine the necessary control action. Likewise, the machine processing agent 350 may pass data regarding the process tool 115 (*e.g.*, idle time, time since last cleaning, tool health, *etc.*) to the controller agent 360. In an alternative embodiment, data regarding the characteristics of the lot 130 and/or process tool 115 may be stored in a centralized data store 360, where it may be accessed by the controller agent 350. The data store 360 may be configured to store data such as include pre-process and post-process metrology data, tool states, lot priorities, *etc.*

The operation of the controller agent 350 is now described in greater detail. The particular control actions taken by the controller agent 350 are implementation specific. For example, if the process tool 115 is a CMP tool, the controller agent 350 may receive pre-polish thickness measurements (*e.g.*, thickness of high features, thickness of low features) and predict a polishing time required to achieve a post-polish target thickness. In the case where the process tool 115 is an etch tool, the controller agent 350 may model the etching performance of the process tool 115 based on pre-etch and/or post-etch thickness measurements. The controller agent 350 may use a control model of the process tool 115 to generate its prediction. The control model may be developed empirically using commonly known linear or non-linear techniques. The control model may be a relatively simple equation based model (*e.g.*, linear, exponential, weighted average, *etc.*) or a more complex model, such as a neural network model, principal component analysis (PCA) model, or a projection to latent structures (PLS) model. The specific implementation of the model may vary depending on the modeling technique selected. Using the control model, the controller agent 350 may determine operating recipe parameters such as etch time, plasma power, temperature, pressure, reactant gas concentrations, *etc.* to reduce post-etch thickness variations. Other control scenarios are possible with other types of process tools 115.

During a configuration of the controller agent 350 (*i.e.*, when it is first instantiated), global configuration variables are defined for use by the controller agent 350. In the illustrated embodiment, these global variables include values of variables from a recipe management system (RMS) (*i.e.*, a global database of recipe settings that stores a default recipe for the current process being implanted by the process tool 115) and context variables. Context variable values define a control thread and typically consist of values for variables such as tool identification code, lot number, operation number, and so forth. In addition, any needed baseline variables are also given values. Examples include an email list for error notifications, values for timeouts, the maximum number of wafers allowed in a lot considered as a "child" lot, previous layer information that the controller agent 350 may use (feed forward information) and so forth.

The controller agent 350 may use the values for lot number and quantity of wafers, as set in the global configuration variables and determine values for lot number, family name, parent name, facility, number of wafers, and status (*i.e.*, whether lot is a parent or a child lot). The previously defined context and RMS information may be used to set the values that the controller agent 350 uses to calculate control moves. For example, the controller agent 350 may use the context information (or "thread" designation) to set the value of a

control model parameter according to the value defined in RMS. A specific example would be setting the value for etch rate used in a control model according to the context of the particular etch chamber and the value for that etch chamber's etch rate as defined in RMS.

In addition, the controller agent 350 may retrieve feed forward metrology information from the data store 360 using a query by lot number and layer name. The controller agent 350 may check elements in an array of data associated with a given lot and fill in default values for missing values. For example, the target of a previous process may be used to set the value of a missing measurement needed as part of the feed forward information used by the controller agent 350. Other methods for setting the value of missing feed forward information, in lieu of using a default value, may also be performed.

The controller agent 350 sets values of the keys and state structures needed for querying the data store 360 to retrieve control states associated with the current control thread. The keys may be used to retrieve the thread state data from the data store 360. The controller agent 350 searches for the thread state data in a stack of ordered data for recent lots processed with this thread context. If such values are found they are passed to a user-defined function containing a control model, which calculates and returns values for the thread states. If no values in the stack are found, the controller agent 350 may search up the hierarchy and retrieve the data from the first hierarchy level that has values for the thread states. The stack and all hierarchy levels are assumed to contain similar data, but at differing degrees of precision.

The controller agent 350 may conduct a jeopardy check by performing a lookup in the data store 360 and retrieving the value for the number of lots in the jeopardy stack (*i.e.*, the stack of lots which were processed on the given thread since the last metrology operation). This value may be compared to a threshold value for number of lots in this jeopardy category, a value typically specified in RMS. If this jeopardy threshold value is not exceeded, the controller agent 350 may continue. If the threshold value is exceeded, the controller agent 350 may abort and send a popup display that instructs an operator to perform a metrology event on a lot from the list of lots in the jeopardy stack prior to determining a control action for the current lot.

The controller agent 350 computes controller inputs (process recipe updates) from the state and target information previously determined, as described above. The controller agent 350 then sends the results of the control action to the machine processing agent 340, which, in turn, updates the operating receipt of the process tool 115 prior to processing the lot 130.

In some embodiments, more than one controller agent 350 may be instantiated for the processing of the lot 130. For example, if the process tool 115 is a photolithography stepper, one controller agent 350 may be invoked to control overlay, and another controller agent 350 may be invoked to control critical dimensions (CD). The multiple controller agents 350 may use feedback from processed wafers to adjust various stepper parameters, such as exposure dose, exposure time, focus, *etc.* In another example, a deposition tool, such as a tool for forming polysilicon layers, may also have multiple controller agents 350 for controlling parameters such as polysilicon grain size and polysilicon layer thickness.

When the controller agents 350 are invoked, the context variables are checked to determine which individual controller agents 350 need to determine control actions. The operation ID indicates the process will be run (*e.g.*, poly gate mask vs. second interlevel dielectric layer mask (ILD)). Each controller agent 350 is only required under certain context situations and will only run if all of those context conditions are met. For

example, the CD controller agent 350 may run for a poly gate mask, but may not run at the second ILD mask process. The overlay controller agent 350, on the other hand, may run at both mask events.

The use of an agent-based control architecture, as described herein, improves overall factory effectiveness and provides more comprehensive automation. More particularly, the autonomous, active software agents 265 schedule and initiate execution of lot scheduling and processing. In general, improved process control reduces product variability, which, in turn, increases product performance and profitability.

The particular embodiments disclosed above are illustrative only, as the invention may be modified and practiced in different but equivalent manners apparent to those skilled in the art having the benefit of the teachings herein. Furthermore, no limitations are intended to the details of construction or design herein shown, other than as described in the claims below. It is therefore evident that the particular embodiments disclosed above may be altered or modified and all such variations are considered within the scope and spirit of the invention. Accordingly, the protection sought herein is as set forth in the claims below.

CLAIMS

1. A manufacturing system (100, 300), comprising:

a process tool (115) configured to process a workpiece (130) in accordance with an operating recipe;

a controller agent (350) configured to determine a control action associated with the processing of the workpiece (130) in the process tool (115); and

a first processing agent (320, 340) associated with at least one of the process tool (115) and the workpiece (130) and being configured to interface with the controller agent (350), receive the control action, and configure the operating recipe based on the control action.

2. The system (100, 300) of claim 1, further comprising a second processing agent (320, 340) associated with the one of the process tool (115) and the workpiece (130) not associated with the first processing agent (320, 340), at least one of the first processing agent (320, 340) and the second processing agent (320, 340) being configured to interface with the controller agent (350).

3. The system (100, 300) of claim 2, wherein the first processing agent (320) is associated with the workpiece (130) and is configured to interface with the controller agent (350) to receive the control action and pass the control action to the second processing agent (340) associated with the process tool (115), the second processing agent (340) being configured to configure the operating recipe based on the control action.

4. The system (100, 300) of claim 1, wherein the controller agent (350) is configured to receive data associated with at least one of the workpiece (130) and the process tool (115) and determine the control action based on a control model and the received data.

5. The system (100, 300) of claim 4, wherein the first processing agent (320) is configured to provide the data associated with the workpiece (130) to the controller agent (350), the second processing agent (340) is configured to provide the data associated with the process tool (115) to the controller agent (350), and the controller agent (350) is configured to determine the control action based on the control model, the data associated with the workpiece (130), and the data associated with the process tool (115).

6. A method for controlling a process tool (115) configured to process a workpiece (130) in accordance with an operating recipe, comprising:

instantiating a first processing agent (320, 340) associated with at least one of the process tool (115) and the workpiece (130);

instantiating a controller agent (350) configured to determine a control action associated with the processing of the workpiece (130) in the process tool (115);

interfacing the first processing agent (320, 340) with the controller agent (350) to receive the control action; and

interfacing the first processing agent (320, 340) and the process tool (115) to configure the operating recipe based on the control action.

7. The method of claim 6, further comprising:

instantiating a second processing agent (320, 340) associated with the one of the process tool (115) and the workpiece (130) not associated with the first processing agent (320, 340); and

interfacing at least one of the first processing agent (320, 340) and the second processing agent (320, 340) with the controller agent (350).

8. The method of claim 7, further comprising:

passing the control action from the first processing agent (320) to the second processing agent (340); and

interfacing the second processing agent (340) and the process tool (115) to configure the operating recipe based on the control action.

9. The method of claim 6, further comprising:

providing data associated with at least one of the workpiece (130) and the process tool (115) to the controller agent (350); and

determining the control action based on a control model and the provided data.

10. The method of claim 9, further comprising:

interfacing the first processing agent (320) with the controller agent (350) to provide the data associated with the workpiece (130);

interfacing the second processing agent (340) with the controller agent (350) to provide the data associated with the process tool (115); and

determining the control action based on the control model and the data associated with the process tool (115) and the workpiece (130).