Axiomatic Design of Machine Control System

Kwangduk D. Lee[1], Nam P. Suh[1] (1), Jae-Hyuk Oh[2]
[1]Massachusetts Institute of Technology, Cambridge, USA
[2]United Technologies Research Center, East Hartford, USA

Abstract
Machine control system development typically relies on the developer's experience and trial-and-error. This ad hoc approach can undermine the very success of the system development, with a lengthy and costly development period and, possibly, an endless cycle of upgrade and maintenance. The resulting system also lacks the interchangeability and reusability. This paper presents the framework based on Axiomatic Design for the systematic design and implementation of machine control systems. It structures a complex control system and guides the design and development of its subcomponents. Various levels of system design issues are addressed in this structural approach. Based on the methodology presented in this paper, a control system for an industrial scale Chemical-Mechanical Polishing (CMP) machine has been developed.

Keywords: Machine Control System, Axiomatic Design, Chemical-Mechanical Polishing

1 INTRODUCTION

Machine control system is a collection of hardware and software, designed to coordinate the output of each individual component to achieve the desired machine functionality. The level of intelligence and the amount of automation required for a modern machine are ever increasing and demanding, powered up by the advances of the computer and the sensing technology. The performance of a machine is often judged by its control system, whether it is a robot, a machine tool, or a microelectronics fabrication equipment. Given the machine hardware, the control system, especially the software part, determines the effectiveness and efficiency of its operation. The control system design is the important part in the overall machine development.

In this era of high competition, it is also important to develop a high performance control system rapidly with the least amount of resources. Minimizing time and capital investment is the top concern in any system development. Hence, we are facing two demanding challenges: how to design a 'best' system to achieve the desired functionality and how to develop such a system in the most economical way? The latter can be called 'the design of the design process', which projects the design efforts on the multiple coordinates of time, capital investment, human resource, etc.

There exist many approaches dealing with the design process. Lu et al.[1] classifies the existing methodologies into three groups. The first group, mainly from the engineering discipline, deals with how a design decision is made to structure the whole system. The second group, largely from business research, views the design process as a dynamic workflow with task dependencies and product information exchange. The CAD and CAE area forms the third group, which views collaborative design process as defining information content and accessibility.

In computer science, various state representation techniques are used to design complex software systems [2].

Despite the considerable contributions from these approaches, there yet exist no comprehensive system design and development tool. As a result, the system design still heavily relies on the experience and know-how of the personnel involved.

This paper presents Axiomatic Design (AD) as a systematic tool to structure a complex machine control system and guide its development. AD was originated as a design decision making tool, but has been expanding its horizon into the design information handling. AD has advanced to create the scientific base for design, which enables designers to make correct design decisions with the highest probability of success [3, 4].

Control system includes both the hardware oriented low-end controllers, such as servo controllers and pressure regulators, and the software oriented high-end controllers, such as command schedulers and motion coordinators. AD will not replace a specific design domain knowledge, such as how to design a PID controller or a signal filter. However, concerning the fact that nowadays the majority of controllers and signal processors are implemented in software with appropriate I/O devices, AD will address the issue of how to realize the designed controller and how to place and structure the individual controllers to ensure the convenience in use and the functional independence. The structuring should provide the abstraction and the independence of the system components. AD is also effective in designing a pure software system [5].

The control system can be divided into three different levels—System, Application, and Function—based on their functionality and scope. This paper will address pertinent issues and show how AD can be employed at each level. System maintenance and upgrade based on AD is also discussed. The CMP machine control system

is presented as a successful example of the Axiomatic control system design, which has been developed by the authors at MIT.

2 CMP MACHINE

Chemical-Mechanical Polishing (CMP) is one of the microelectronics fabrication processes, in which a layer deposited on a silicon wafer is planarized by the means of mechanical polishing with the assistance of chemical agents. A layer is planarized either to expose the underlying circuitry or to provide a flat base layer for subsequent processes, such as the lithography [6].

The CMP machine has been developed at MIT by the CMP research group to meet the growing research needs. AD was employed to establish its system architecture (process requirement, machine hardware, control system, etc.). It is designed to enable multi-step polishing of 200 mm wafers, expandable to 300 mm generation. As a consequence, the platform is large (2.7 × 1.5 × 2 m) with numerous components to be controlled and coordinated. Figure 1 shows the drawing of the machine.

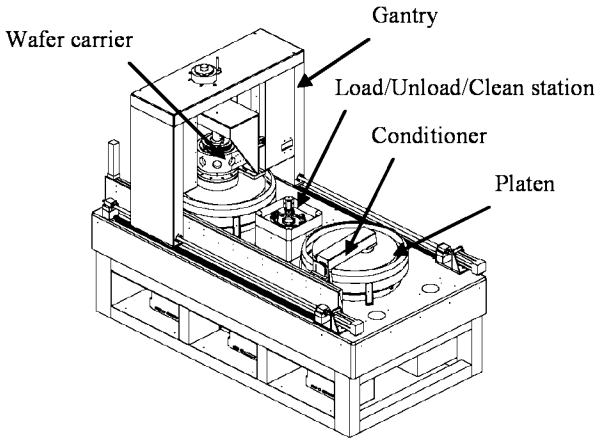


Figure1: CMP machine.

The machine has 9 servo motors, 4 pumps, 8 pressure regulators and more than 60 on/off type actuators and numerous sensors. The control system requires power wiring and signal interfacing of each components, signal processing, low-level controllers, a coordinator for various component controllers, a scheduler to dispatch commands in sequences to process wafers following a 'recipe,' a user interface, a recipe editor, etc. A systematic structuring tool is required to initiate the design.

Axiomatic Design is inherently hierarchical due to the mapping between the different domains and the subsequent decomposition to yield the working level design solutions. AD will be used to design this complex machine control system.

3 SYSTEM LEVEL DESIGN

At the initial stage of a system development, it is often difficult to specify what is necessary to constitute the system, or more precisely what is required for the system to function. In AD, it belongs to the realm of the high level decomposition. At this level, AD is used to organize and orient the overall system design, by clearly stating Functional Requirements (FRs). Design Parameters (DPs) at this stage do not have specific details, but required for the further system decomposition. Suppose we conceived an FR stating "Control the machine in an automatic manner so that wafers can be processed in a preprogrammed way with minimal human intervention." The corresponding DP will simply be "Auto mode." No specific detail needs to be known at this stage. The subsequent decompositions will clarify the auto mode.

Stating FRs at the initial design stage serves the purpose of system definition and functionality clarification. The DP is a mere conceptualization of the corresponding FR, yet enabling further system development by the decompositions to follow. The strength of AD lies in the fact that it requests the designer to come up with a clearly defined set of FRs, which leads to the strategic placement of necessary system components.

Axiomatic Design at system level focuses on identifying the major system components and transforms the complex and poorly-acknowledged original system to a hierarchical set of related subsystems which becomes much easier to design and implement. The system level design of the CMP machine control system will be given to illustrate this point.

Axiomatic decomposition often starts with a single objective or mission statement. Table 1 shows the mission statement, which also serves the highest level FR. The equipment control system is a conceptualization of the FR statement- a clear, yet extensive definition of the problem. A few more decompositions will lead to the DPs with more specific details.

Table 1: Mission statement (highest level FR).

FR	DP
Coordinate the individual components of the CMP machine in a systematic manner so that it can process wafers effectively and efficiently to the desired specifications	Equipment control system

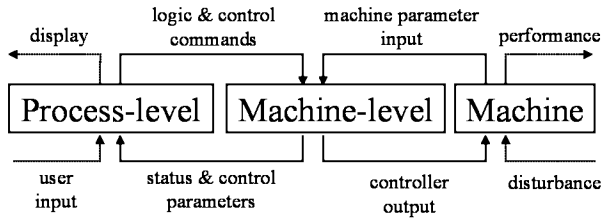
The equipment control system is essentially a set of software designed to achieve the goal of coordinated wafer processing, with appropriate interfaces to the machine and the user. Here, the control means both the control of the hardware components, such as position servos, and the control of these controllers to obtain the desired set of actions. We may decompose the system based on the two conceived functionalities, as shown in Table 2.

Table 2: Decomposition of mission statement.

	FR	DP
1	Generate control actions from each individual machine components	Machine-level control system
2	Organize the controlled outputs for wafer processing	Process-level control system

$$\begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} X & O \\ X & X \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix}$$

The design is decoupled, because the design of a process depends on what can be done on the machine side. We may add a third DP, namely 'Operation-level control system' in a highly integrated and automated environment to deal with the cooperation with the supportive equipments and the communication with a higher level production management system. These systems interact with each other and as a whole handles the information flow between the user and the machine hardware. Figure 2 shows the interaction between the systems.



* Logic commands: function buttons, initialization
* Control commands: position, velocity, pressure, flow rate, valve and switch on/off, etc.

* Status parameters: error, operative status
* Control parameters: position, velocity, sensor output, time, etc.

Figure 2: Information flow between the systems.

Subsequent decompositions reveal the content of each system. For example, the Machine-level control system can be further decomposed as shown in Table 3.

Table 3: Machine-level control system.

	FR1.x	DP1.x
1	Receive inputs and send outputs	I/O unit
2	Process incoming signals	Signal processing unit
3	Produce controlled outputs	Controller unit
4	Structure the Machine-level control system	Machine-level overhead

$$\begin{Bmatrix} FR11 \\ FR12 \\ FR13 \\ FR14 \end{Bmatrix} = \begin{bmatrix} X & O & O & O \\ X & X & O & O \\ X & X & X & O \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} DP11 \\ DP12 \\ DP13 \\ DP14 \end{Bmatrix}$$

The I/O unit handles the signal flow in and out of the Machine-level, and communicates both with the machine and the Process-level. The signal processing and the controller unit are the core working modules. Concerning the amount of work to be performed by these units, the Machine-level overhead serves the role of an administrator, organizing individual unit actions and coordinating the time and space related issues.

After the system level decomposition, the DPs begin to have specific meanings and tasks, and seem to be designable from now on. The I/O unit design, for example, will start to look at what kinds of signals are required to control the machine, then what types of I/O cards are required for the external processing system and what types of I/O variables are used for those cards, etc. AD at the system level has identified and specified the major building blocks of the system.

The Process-level Control System can also be decomposed into the supportive unit which includes the recipe builder, the equipment database and the set up mode, and the process unit which includes the manual and the auto mode.

4 APPLICATION LEVEL DESIGN

An application is a subsystem of the parent system, which is identifiable as a unit due to its clearly defined functionality, yet has a variety of outputs compared to a function which has a single dedicated output. For instance, the auto mode and the recipe builder are the application level components of the equipment control system. In this section, we will examine how AD can be employed to design the auto mode as an example of the Application Level design.

The FRs are well defined at this stage, due to the foregoing decompositions at the higher level. More attention is paid to selecting appropriate DPs to satisfy the established FRs. From now on, the selection and design of each DP has technical flavors. For example, to design a graphical user interface, the programmer should design backend processes first to process data, next frontend window layout, then selection of programming languages, etc.

The application level has clearly defined FRs and the issue here is how to select and substantiate the right DPs. It is often required for the designer to have a knowledge and experience related to the specific domain in which the design is being performed. However, AD is employed to prevent the pitfalls of heuristic and trial-and-error approach of experienced persons and guide to the 'betterment.'

Table 4: Decomposition of auto mode.

	FR222.x	DP222.x
1	Structure the auto mode software	Auto mode overhead
2	Access recipe files	Recipe link
3	Interact with a user	User interface
4	Communicate with the Machine-level	Machine-level interface
5	Designate distinct steps for automatic wafer processing	Process steps

$$\begin{Bmatrix} FR2221 \\ FR2222 \\ FR2223 \\ FR2224 \\ FR2225 \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & X & X & O & O \\ X & O & O & X & O \\ X & O & O & O & X \end{bmatrix} \begin{Bmatrix} DP2221 \\ DP2222 \\ DP2223 \\ DP2224 \\ DP2225 \end{Bmatrix}$$

The auto mode is decomposed into five DPs as shown in Table 4. The auto mode overhead organizes the whole auto mode structure and coordinates four other DPs. The recipe link loads and unloads the recipe, edited and saved by the recipe editor. The user interface handles the interaction between the user and the auto mode. The machine-level interface provides the communication channel between the auto mode and the machine. These three DPs function as the I/O stream of the auto mode. The process steps are the preprogrammed autonomous stages to support the recipe based automatic wafer processing. In conjunction with the process parameters downloaded from a recipe file, the sequential execution of the steps induces the intended physical change of the wafers.

Each DP can be decomposed further to refine its content. Table 5 shows the decomposition of the auto mode overhead. It specifies the I/O frequencies of the auto mode, controls the data flow between various application components, and performs the non-process routines such as initialization and termination.

DP22211: I/O frequency specifies the intervals of the inputs and the outputs of the auto mode. Every I/O stream has a predetermined interval, except the user input which occurs at a user generated event. 18 Hz PC timer is used as a clock device.

DP22212: User input handler extracts logic commands (Run, Stop, etc.) from the user interface and passes them to the Machine-level output handler.

DP22213: Recipe handler extracts the process parameters from the recipe link for process steps.

DP22214: Machine-level output handler sends the data packets and the logic commands to the Machine-level interface, which sends them to the Machine-level control system.

Table 5: Decomposition of auto mode overhead.

	FR2221.x	DP2221.x
1	Designate how often the auto mode inputs and outputs are updated	I/O frequency
2	Process user input	User input handler
3	Process recipe data	Recipe handler
4	Send outputs to the Machine-level	Machine-level output handler
5	Process inputs from the Machine-level	Machine-level input handler
6	Update the process information to the user	User display handler
7	Initialize the auto mode	Initialization procedure
8	Terminate the auto mode	Termination procedure

$$\begin{Bmatrix} FR22211 \\ FR22212 \\ FR22213 \\ FR22214 \\ FR22215 \\ FR22216 \\ FR22217 \\ FR22218 \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O & O & O & O \\ X & X & O & O & O & O & O & O \\ X & X & X & O & O & O & O & O \\ X & X & X & X & O & O & O & O \\ X & O & O & O & X & O & O & O \\ X & O & X & O & X & X & O & O \\ X & O & O & X & X & O & X & O \\ O & O & O & O & O & O & O & X \end{bmatrix} \begin{Bmatrix} DP22211 \\ DP22212 \\ DP22213 \\ DP22214 \\ DP22215 \\ DP22216 \\ DP22217 \\ DP22218 \end{Bmatrix}$$

DP22215: Machine-level input handler receives the parameter and status inputs from the Machine-level control system via the Machine-level interface. The handler unpacks the inputs, analyzes them, and assigns them to the user display handler.

DP22216: User display handler passes the Machine-level inputs to the designated sections of the user interface window.

DP22217: Auto mode initialization procedure is executed when the auto mode is loaded to the Process-level control system. It assigns the constants, initializes the variables, and calls the default routine to bring the machine to the ready status.

Table 6: Decomposition of process steps.

	FR2225.x	DP2225.x
1	Create sets of control actions commonly used by the steps	Sub-steps
2	Provide a default waiting step in automatic wafer processing	Step_Ready
3	Condition polishing pads	Step_Condition
4	Load a wafer	Step_Load
5	Polish a wafer	Step_Polish
6	Clean a wafer	Step_Clean
7	Unload a wafer	Step_Unload
8	Clean the conditioner	Step_CondClean

$$\begin{Bmatrix} FR22251 \\ FR22252 \\ FR22253 \\ FR22254 \\ FR22255 \\ FR22256 \\ FR22257 \\ FR22258 \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O & O & O & O \\ X & X & O & O & O & O & O & O \\ X & O & X & O & O & O & O & O \\ X & O & X & X & O & O & O & O \\ X & O & X & O & X & O & O & O \\ X & O & X & O & O & X & O & O \\ X & O & X & O & O & O & X & O \\ X & O & O & O & O & O & O & X \end{bmatrix} \begin{Bmatrix} DP22251 \\ DP22252 \\ DP22253 \\ DP22254 \\ DP22255 \\ DP22256 \\ DP22257 \\ DP22258 \end{Bmatrix}$$

DP22218: Auto mode termination procedure can be called either to switch to the manual mode or to terminate the machine operation. In either case, the auto mode user interface is unloaded from the operator console.

Processing of a wafer demands hundreds of individual control actions to take place both serial and parallel. Thus it is necessary to organize these actions into certain meaningful groups. Small sets of control actions can be organized as a 'sub-step' to be used by a bigger 'step.' A process step is a sequence of control actions and sub-steps. A step is distinguished and designed as a building block in constructing a recipe. Then a user can construct his or her own process recipe by placing the individual steps in a desired sequence with the edited step parameters. Table 6 shows the decomposition.

Other DPs of the auto mode can be constructed in a similar fashion and implemented. Figure 3 shows the user interface window of the operator console when booted as the auto mode.

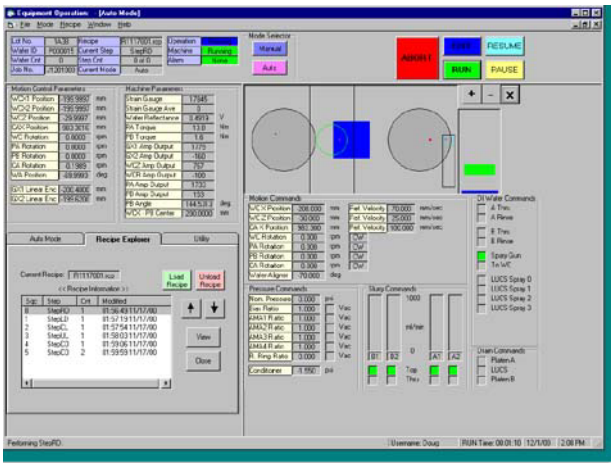


Figure 3: Auto mode user interface.

5 FUNCTION LEVEL DESIGN

It is often necessary to design a specific sequence while designing an application. The sequences usually comprise the lowest level of the system decomposition (leaf modules). Thus if an engineer designs a specific sequence, he or she is near the completion of the whole system design.

If the sequential stages are put into the form of design equation, they usually become decoupled, because of their ordered nature. But a stage can be uncoupled from its immediately previous ones, implying that it can be a parallel procedure to them. The information contained in the design matrix can easily be exported to construct a sequential functional diagram, which graphically represents the sequential algorithm. Eppinger et al. [7] also discusses the use of the matrix representation to reveal the sequential nature of the design process.

AD is a powerful tool in designing the logic sequence of a function. A function is a set of procedures, which has a single dedicated output. In this section, AD is employed to design the sequences.

Step_Load (DP22254) requires a sequential algorithm to guide individual control actions to the common goal of coordinated wafer loading. Table 7 shows the decomposition of the algorithm.

Logical requirements of the step are expressed as the FRs and the DPs are chosen to realize the corresponding stages, which are mostly sub-steps and

procedures. Each procedure has a single purpose and can be coded with a few lines in the auto mode software.

Table 7: Sequential algorithm (Step_Load).

	FR222543.x	DP222543.x
1	Move WC to the center and the clearance height	WC move procedure
2	Move WC down to the pick up position	WC move procedure
3	Align wafer	Wafer alignment procedure
4	Pressurize WC chambers	WC pressurization procedure
5	Apply chamber vacuum	Chamber vacuuming procedure
6	Apply bias vacuum	Bias vacuuming procedure
7	Move WC up slowly to the clearance height	WC move procedure
8	Open wafer aligner	Aligner open procedure
9	Move WC to final position	WC move procedure
A	Allow manual loading, if wafer pickup failed	Wafer sensing and handling procedure

$$\begin{Bmatrix} FR2225431 \\ FR2225432 \\ FR2225433 \\ FR2225434 \\ FR2225435 \\ FR2225436 \\ FR2225437 \\ FR2225438 \\ FR2225439 \\ FR222543A \end{Bmatrix} = \begin{bmatrix} X & O & O & O & O & O & O & O & O & O \\ X & X & O & O & O & O & O & O & O & O \\ X & O & X & O & O & O & O & O & O & O \\ X & X & X & X & O & O & O & O & O & O \\ X & X & X & X & X & O & O & O & O & O \\ X & X & X & X & X & X & O & O & O & O \\ X & X & X & X & X & X & X & O & O & O \\ X & X & X & X & X & X & O & X & O & O \\ X & X & X & X & X & X & X & X & X & O \\ X & X & X & X & X & X & X & X & X & X \end{bmatrix} \begin{Bmatrix} DP2225431 \\ DP2225432 \\ DP2225433 \\ DP2225434 \\ DP2225435 \\ DP2225436 \\ DP2225437 \\ DP2225438 \\ DP2225439 \\ DP222543A \end{Bmatrix}$$

uncoupled

The examination of the causal relationship from the design matrix reveals two uncoupled elements in the lower diagonal, although the rest of them are decoupled. Decoupling is natural, because in a sequential algorithm the subsequent stages can not be executed without the occurrence of the previous ones. However, the uncoupling means the corresponding stage can occur in parallel to its immediately preceding one. For example, the design matrix reveals that DP222543.3 can be a parallel process to DP222543.2. Utilizing this information, the sequential functional diagram can easily be constructed.

Figure 4 shows the sequential functional diagram constructed from the design matrix. Each FR/DP set corresponds to a stage in the sequential algorithm. The transition conditions between the stages are represented by the corresponding process variables. At the same time, the evolution or progress of the sequence is recorded by PV222541, which is the status indicator of the loading step. A higher level program watches the internal stage of the step and takes corresponding actions based on the monitoring. The logic flow is branched into parallel processes, where necessary, and joined again at an '⊗ (AND)' junction.

6 EVALUATION, MAINTENANCE AND UPGRADE

Once the design is completed, individual components are implemented and tested before being assembled into a system. The performance of the system as a whole is also tested. The AD tables and matrices are used to guide the evaluation procedure.

After the low-end controllers on the Machine-level control system and the communication handlers between the two levels are tested, the Step_LD of the Process-level control system can be tested. The engineer will first check if each stage is correctly executed and its transition condition is met as specified in the AD table and matrix, and if the required accuracy of each physical actuation is achieved. The step monitoring and supervision of the Process-level control system is also checked, by executing a series of process steps in addition to the loading step. Ultimately, the Step_LD will be tested against the loading reliablility (for example, 99.7% of probability of success in loading wafers).

AD is useful not only for the initial design itself, but also for the troubleshooting and the subsequent modification.

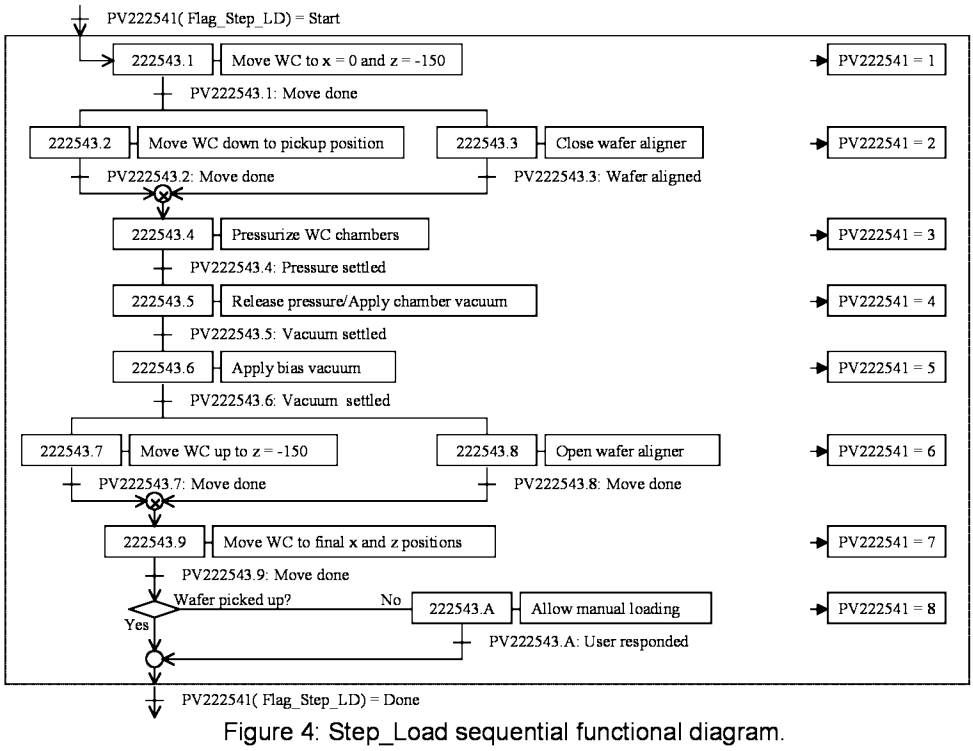


Figure 4: Step_Load sequential functional diagram.

For these maintenance and upgrade purposes, the AD tables and matrices also serve the role of system documentation, which contains the description and logic of the system architecture.

Troubleshooting may be required either during the initial evaluation period or during the maintenance period after the design is completed.

For example, suppose the machine repeatedly fails to pick up the wafers. An engineer will be assigned to solve the problem. He or she is not necessrily the person who originally designed the Step_LD (and the control system). Assuming no prior knowledge, he or she will first examine the design table and decide that the sequential stages from DP222543.1 to DP222543.7 should be checked, because the physical wafer pick up occurs at DP222543.7. The effectiveness of the physical actuation at each stage will be examined first, such as the precision in the wafer carrier movement, the acceleration in the pick up movement, and the operation of valves in pressurization and vacuuming procedures. If no problem is detected, process parameters can be checked, such as the amount of vacuum and its settling time. Further investigation may lead to the examination of the sequential functional execution of the control software and eventually the sequential logic itself. In this manner, the troubleshooting becomes a logical and systematic procedure, stemming from the AD table and matrix.

Modification or upgrade can occur after the system development, at various levels. On one extreme, to upgrade the machine to the commercial grade, we may want to add the Operation-level control system to prepare for the automation and system integration at the customer site. On the other hand, we may simply want to add a few more stages in the Step_LD to enhance the loading reliability. In either case, the AD tables and matrices are first consulted, and how each component should be designed and its effect to the rest of the system are studied using the design matrices. For example, the Operation-level control system should utilize and communicate with the existing components of the current control system and should not create constrains to the lower-level systems.

In another case, suppose, due to the change in the customer requirement, the loading step has to read the ID (bar code) of the wafer before loading it. An engineener will be asked to insert a scanning stage to the loading step. He or she will simply look at the Step_Load design table and logically decide that the scanning stage should be inserted after the wafer alignment but before the WC pressurization. The new wafer ID reading is designated as FR/DP222543.4 and the subsequent stages are shifted by one. The engineer may also recommend to modify the wafer alignment procedure to provide the required precision for the scanner head.

7 CONCLUSION

This paper presents the design of machine control system based on Axiomatic Design. AD is an effective tool in control system design, because it logically identifies the essential system components and structures the once complex set of requirements and ideas into the coherent set of FR/DP architectures. AD leads to the betterment of the system design and ensures the effectiveness and the efficiency both in the product itself and its development.

The system level design requires a clear statement of FRs, with the conceptualizing DPs. The FR definition at this level serves the purpose of system organization and structuring.

The application level design starts with the clearly defined FR set, from the system level. In many cases, specific knowledge is required to select and 'design' the optimum DPs. AD is useful to break down the huge web of functionalities into an organic set of individual functions, which are easy to design.

A specific function of an application is designable, because of its clearly defined purpose and the well established input and output relationship. However, AD can still be used to design the sequential algorithm of a function, which is encountered frequently in control system design. The information contained in the design matrix is easily exported to construct the sequential functional diagram.

The AD system architecture and the accompanying desing matrices and tables eases the tasks of system evaluation, maintenance and upgrade. Concerning the complexity of the modern machine control system, the information contained in the AD system architecture is essential to guide tests, troubleshootings and modifications in a fail-safe way, withoug passing 'bugs'.

Employing these methodologies, the authors have successfully developed the control system for an industrial scale CMP machine. It took less than a year for the authors to build the system from the scratch. The resulting system is fully functional without any major error or drawback. This is the complete example of how AD can be used to design an industrial scale control system, beyond the court of the academia.

Although this paper has presented the design of a machine control system, the methodologies proposed here can be applied to the design of any type of control system, such as the automotive, the aircraft, the traffic and the factory management control system.

8 REFERENCES

[1] S. C.-Y. Lu, J. Cai, W. Burkett and F. Udwadia, 2000, A Methodology for Collaborative Design Process and Conflict Analysis, Annals of CIRP, Vol. 49, No. 1, pp 69-73.

[2] D. Harel, 1987, Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, Vol. 8, pp 231-274.

[3] N. P. Suh, 1990, The Principles of Design, Oxford University Press, New York, NY.

[4] N. P. Suh, 2001, Axiomatic Design: Advances and Applications, to be published by Oxford University Press, New York, NY.

[5] N. P. Suh and S. H. Do, 2000, Axiomatic Design of Software Systems, Annals of CIRP, Vol. 49, No. 1, pp 95-100.

[6] R. DeJule, 1997, CMP Challenges, Semiconductor International, November, pp 55-60.

[7] S. D. Eppinger, D. E. Whitney, R. P. Smith and D. A. Gebala, 1994, A Model-Based Method for Organizing Tasks in Product Development, Research in Engineering Design, Vol. 6, pp 1-13.