THE OPEN ARCHITECTURE SCHEDULING SYSTEM FOR A SINGLE-ARMED CLUSTER TOOL WITH PM CLEANING OPERATIONS

Dong-Hyun Roh[(a)]**, Tae-Eog Lee**[(b)]

[(a),(b)] Department of Industrial & Systems Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea

[(a)]rdh@simlab.kaist.ac.kr, [(b)]telee@kaist.ac.kr

ABSTRACT

Cluster tools are a type of widely used semiconductor manufacturing equipment. Generally, a cluster tool is operated on a built-in schedule; however, it is impossible to modify or change the built-in schedule because of the closed architecture of the scheduling system for the tool. In this study, we propose a framework for an open architecture scheduling system for a single-armed cluster tool with PM cleaning operations. The scheduling system works by scheduling command files that can be modified or replaced from outside. As an application of the framework, performance comparison analysis between the backward and backward(z) sequences in a single-armed cluster tool with multi-period PM cleaning operations is conducted.

Keywords: cluster tool, open architecture scheduling system, virtual cluster tool, PM cleaning operation

1. INTRODUCTION

In modern semiconductor manufacturing systems, about 300 processes are required for the fabrication of a semiconductor product. Most processes, including etching, vapor deposition, and wafer cleaning, are performed using cluster tools (Yu and Lee 2017). Cluster tools are the most popular type of configurable semiconductor manufacturing equipment; they consist of several single-wafer process modules (PMs) and a material-handling robot called a transporting module (TM). PMs are modular and can be detached and attached; TM is responsible for transporting wafers. Usually, a cluster tool is operated according to a specified wafer flow pattern and several operational constraints. There have been numerous studies on the operation of cluster tools for various wafer flow patterns and constraints (Venkatesh et al. 1997; Geismar 2004; Lee et al. 2014; Wu et al. 2011; Kim et al. 2015; Kim et al. 2013; Kim et al. 2013; Yu and Lee 2017; Rostami et al 2001; Kim et al. 2003; Wu 2008).

Cluster tools have diverse structures depending on the types of PM and TM; structures include single-armed and dual-armed cluster tools, multi-slot cluster tools, and in-line multi-cluster tools (Lee 2008). Inter alia, the most commonly used structure is a radial cluster tool, in which PMs radially surround the tool. A radial cluster tool is called a single-armed or dual-armed cluster tool according to the number of robot grippers. It is known that the backward sequence and the swap sequence are optimal robot task sequences for the single-armed and dual-armed cluster tools with series-parallel wafer flow patterns, respectively (Lee 2008).

Generally, a semiconductor fabrication plant, a so-called "fab", purchases semiconductor manufacturing equipment including cluster tools from a variety of equipment manufacturing companies. Therefore, most cluster tools operate on the schedule built into the tool by the equipment manufacturing company. The built-in schedule usually consists of a benchmark schedule mainly based on the backward or the swap sequence and several exception-handling techniques.

However, there exists an issue that built-in schedules and tool schedulers are different for each equipment manufacturing company. In addition, a tool scheduler has a closed architecture; in other words, only the equipment manufacturing company can modify the internal structure and operating logic of the scheduler. This means that a fab cannot arbitrarily modify a tool schedule to induce high productivity and exception-handling techniques for quality control. Since, in recent years, the method of semiconductor production has changed from the existing high-volume low-mix production to high-mix low-volume production, a tool scheduler with closed architecture leads to difficulty in optimizing the operation of the tool in situations in which frequent schedule changes are needed. Furthermore, there also exists a disadvantage that it is difficult to investigate whether the existing built-in schedule provides optimal productivity or, if not, which parts need to be improved.

Therefore, a system capable of changing a tool schedule from the outside as desired, called an open architecture scheduling system, is needed. In order to implement the open architecture scheduling system, a standard scheduling command scheme of cluster tools that all equipment manufacturing companies and fabs can understand is needed. If all cluster tool schedulers follow the same scheduling command scheme, it becomes quite easy for a fab to replace the old tool schedule with a new one. This idea is similar to the NC programming language or the NC code used in CNC systems. Since the NC code has a standardized command scheme, a work schedule of a CNC machine can be easily changed from outside, even though the NC code is generated using a variety of CAD/CAM software.

However, semiconductor manufacturing equipment, including cluster tools, is much more sophisticated and has diverse structures and operational constraints depending on the processes. In addition, even equipment that performs the same process may have different structures depending on the equipment manufacturing company. Therefore, establishing a standard scheduling command scheme for cluster tools requires much expert effort in the industrial setting.

For this reason, this study proposes a framework for an open architecture scheduling system. As an example, we develop a virtual cluster tool (VCT) for a single-armed cluster tool with PM cleaning operations and apply the proposed framework to describe how the open architecture scheduling system works. The framework can be extended to cluster tools of various architectures.

In this paper, we first describe the open architecture scheduling system framework. The framework improves the basic concepts of the open scheduling architecture of the cluster tool proposed by Lee and Lee (2010). Then, we define the structure and operation of a single-armed cluster tool with PM cleaning operations. To apply the open architecture scheduling system framework, we also suggest a reference model of the VCT. As an application, we conduct a performance analysis for the backward and the backward(z) sequences suggested by Yu and Lee (2017).

2. ARCHITECTURE AND OPERATION OF A SINGLE-ARMED CLUSTER TOOL WITH PM CLEANING OPERATIONS

2.1. Architecture of a Single-armed Cluster Tool

Cluster tools have a variety of structures. Among them, the single-armed cluster tool having a single-armed transporting module and radially located PMs is one of the most used pieces of manufacturing equipment in leading fabs. A single-armed cluster tool consists of a TM with a single gripper, several, usually two to six, PMs, two loadlock modules for loading a wafer cassette, and an aligner module for aligning a wafer unloaded from a loadlock (Lee 2008). Among the components, the TM and PMs are the most important for the operation.

2.2. Operation of a Single-armed Cluster Tool

When a cassette containing 25 wafers is loaded into a loadlock, a TM carries a wafer to the PMs one by one. The TM only performs wafer loading into the PM, wafer unloading from the PM, and moving operations. In addition, TM can perform one job at a time.

Each wafer is processed according to a predetermined recipe, which defines the sequence of process steps that the wafer should visit and the process time for each process step. For each process step, to raise productivity, several PMs are assigned as parallel PMs. When a wafer is processed sequentially in a tool having parallel PMs, the wafer flow pattern is referred to as a series-parallel flow pattern. After all processes have been completed, the wafer is returned to the loadlock. This process continues until all wafers in the cassette have been processed.

In order for a wafer to be loaded into a PM, the chuck that serves to fix the wafer entering the PM should first be prepared. When the chuck is ready, the slit valve door opens and the TM loads the wafer. After wafer loading, the slit valve door is closed and the chuck firmly holds the wafer. Depending on the type of PM, pumping and venting operations may be required to prepare the process.

Therefore, the number of process steps, the number of parallel PMs for each process step, the process time for each PM, the required times for TM operations (loading, unloading, and moving), and the required task times of the slit valve door and chuck operations for each PM are needed to define the configuration of a single-armed cluster tool with a series-parallel flow pattern.

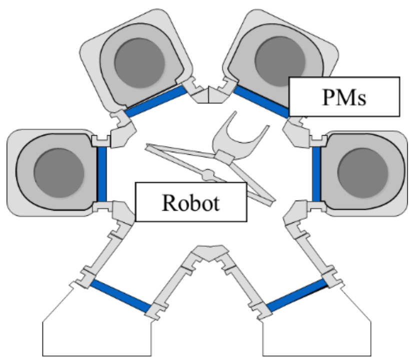

Figure 1: Single-armed Cluster Tool

2.3. Periodic PM Cleaning Operations

As the diameter of a wafer increases and the thickness of the wafer circuit becomes thinner, enhanced wafer quality control is required. Therefore, it becomes necessary to remove residual chemicals in the PM after the wafer process completes. A PM cleaning operation is a process of removing these impurities (Kim et al. 2013; Yu and Lee 2017). It is performed after a predefined number of wafer processes, which is called a cleaning period. During the cleaning operation, a wafer cannot be loaded.

In terms of operation, a cluster tool with periodic PM cleaning operations has differences from a typical single-armed cluster tool. To define an operation of a tool with periodic PM cleaning operations, the cleaning period and the cleaning time for each PM are additionally required. Consequently, the tool configuration is defined by information describing the tool operation which is shown in Table 1.

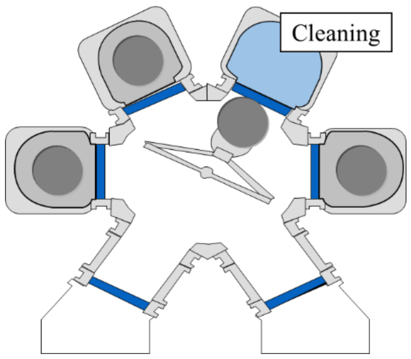

Figure 2: Single-armed Cluster Tool with PM Cleaning Operations

Table 1: Information Required to Define Tool Configuration

Type	Information Required to Define Tool Operation
Typical Single-armed Cluster Tool	Number of Process Steps
	Number of Parallel PMs for Each Step
	Process Time for Each PM
	Required Task Times for TM Operations (Loading, Unloading, Moving)
	Required Task Times for Slit Valve and Chuck
with PM Cleaning Operations	Cleaning Time for Each PM
	Cleaning Period for Each PM

3. OPEN ARCHITECTURE SCHEDULING SYSTEM OF SINGLE-ARMED CLUSTER TOOL WITH PM CLEANING OPERATIONS

The open architecture scheduling system of the cluster tool is a tool scheduler that can receive from the outside information necessary for scheduling. However, existing cluster tool controllers and schedulers have been developed independently for each equipment manufacturing company and it is not easy to implement them because there is no standardized scheduling model or rule. Therefore, a standard for the cluster tool scheduling commands should be given priority. It is difficult to standardize the scheduling commands of all cluster tools because of the various types of cluster tools depending on their usage, function, and required performance level. Furthermore, there exist differences in tools due to technology gaps among manufacturing companies. Consequently, this study suggests an open architecture scheduling system for a single-arm cluster tool with PM cleaning operations.

Scheduling a cluster tool requires three pieces of information: the tool architecture information, the benchmark TM task schedules, and exception-handling techniques. The tool architecture information is the information about the tool configuration and the tool operation, listed in Table 1. The benchmark TM task schedules refer to the work schedule of a TM when the cluster tool is in a steady state. A TM task schedule consists of a TM task sequence and a timing rule. Lastly, exception-handling techniques refer to TM task schedules when a tool is in transient state or when an exception occurs. The types of exceptions, criteria, and coping methods are included.

Lee and Lee (2010) suggested an XML-based file formats that contains integrated scheduling information, called the Scheduling Command File (SCF). In this study, we subdivide the SCF into three categories so that the schedule can be more effectively adjusted. Using SCFs, scheduling is performed by referencing the information from each SCF at each necessary moment. Each SCF can be replaced or modified as needed. Therefore, proposing standardized SCFs is the most important task in implementing the open architecture scheduling system. We briefly describe the information contained in each SCF and how the scheduler refers to each SCF.

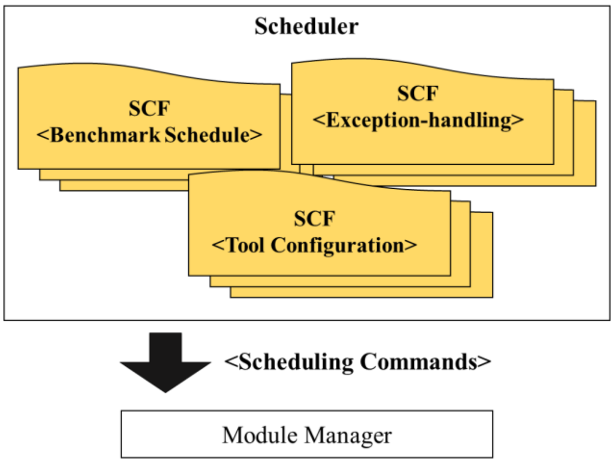

Figure 3: Scheduling Command Files in a Tool Scheduler

3.1. Tool Architecture Information SCF

In order to determine a tool schedule, the tool scheduler must first have information on the tool configuration. It also must have information on the constraints that must be met in the operation of the tool. A ‘tool architecture information SCF’ contains the information on tool configuration and operational constraints. In this study, the information is listed in Table 1.

3.2. Benchmark TM Task Schedules SCF

Most previous studies have expressed a benchmark schedule through a Petri net or a timed event graph (TEG) (Lee 2008). Therefore a ‘benchmark TM task schedules SCF’ contains the information for the Petri net model of the schedule, usually expressed by an incidence matrix and a token marking vector, the firing sequence, and the corresponding firing rule.

Among numerous benchmark TM task schedules, the most commonly used schedule in the single-armed cluster tool is the backward sequence and the earliest starting rule (Lee 2008). The earliest starting rule lets a TM perform a task as soon as possible.

3.3. Exception-handling Techniques SCF

An ‘exception-handling techniques SCF’ contains information about types of exceptions that can be controlled, a precise criteria for each exception, and a TM schedule for each exception. In the types of exceptions, not only hardware problems such as a PM or a TM failure but also problems in a tool schedule such as transient periods of the tool or a K-periodic schedule are included. As cluster tools become increasingly sophisticated, more exceptions arise. There are also many cases in which the exceptions occur in combination. Therefore, it becomes an issue to establish precise criteria for all exceptions and corresponding coping methods.

4. DEVELOPMENT OF A VIRTUAL SINGLE-ARMED CLUSTER TOOL WITH PERIODIC PM CLEANING OPERATIONS

The open architecture scheduling system proposed in this study is a system capable of operating a tool through an external input schedule; thus, a fab can arbitrarily adjust the schedule. However, there exist significant time and money costs to apply the system to a real cluster tool. Therefore, we develop a VCT that operates in a manner similar to the actual cluster tool and apply the framework to the VCT. Among numerous studies about VCTs (Shin et al. 2001; Joo and Lee 2004; Kim and Lee 2013; Min and Lin 2013; Niedermayer and Rose 2003; Pan and Bao 2012), we implemented the virtual cluster tool based on the VCT framework suggested by Joo and Lee (2004).

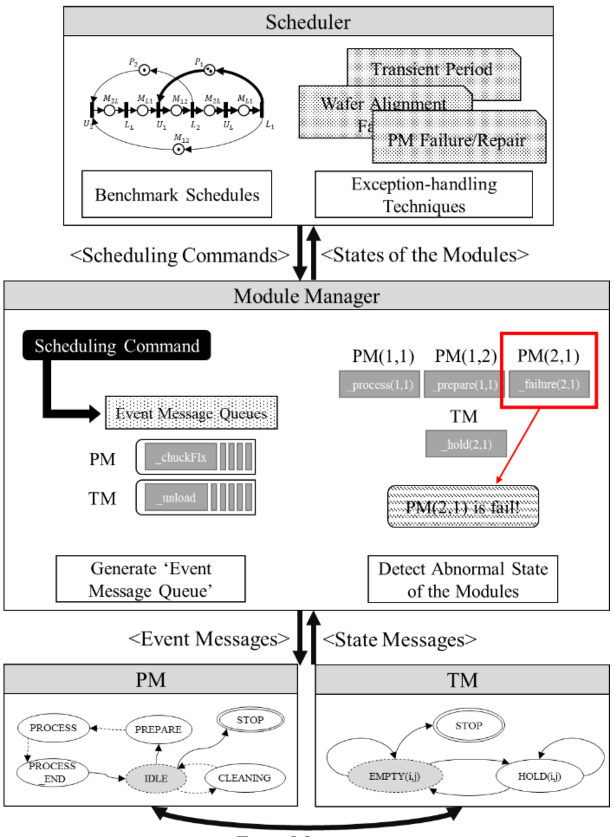

Figure 4: Reference Model of the VCT

4.1. Reference Model of the VCT

The core components of the cluster tool are PMs, a TM, the tool scheduler, and the module manager. The PM and the TM are devices that process and transport wafers; the tool scheduler is responsible for scheduling tasks for the modules, especially for the TM. The module manager is a kind of interface module that interprets scheduling commands coming from the scheduler into a language understood by the modules and transfers the messages to each module.

The VCT also consists of PMs, a TM, the tool scheduler, and the module manager; the components in the VCT are modeled to perform the same functions as the components of the real cluster tool. Fig. 4 demonstrates the reference model of the VCT. The reference model shows what types of messages are exchanged between the components and what functions and information each component has.

4.2. Modeling of Cluster Tool Components

In order to implement the VCT, it is necessary to accurately model the internal operating logic of each component. In this study, we model the internal operating logics of the PMs and the TM as state graphs. Brief explanations of the tool scheduler and the module manager are also included.

4.2.1. State Graph Models

We first briefly explain the state graph model. A state graph is an extended version of timed automata having state variables, system variables, and timers (Choi and Kang 2013). In addition, a state graph allows transition conditions and three types of actions: entry, input, and transition actions. In the graph, "?" denotes an input event or message, and "!" denotes an output event or message. A transition condition is denoted by "~". Among the states of the graph, the initial state and the final state are expressed differently. Fig. 5 shows an example of a simple state graph.

Each state has its own name, entry actions, and a timer [$\Delta(t_0)$]. A state graph model can be described by a state transition table. In a state transition table, all components for constructing the state graph model are specified. The state transition table shown in Table 2 represents the state graph model in Fig. 5.

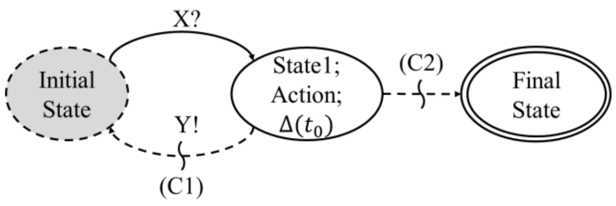

Figure 5: Example of State Graph Model

Table 2: State Transition Table of Model in Fig. 5

State			Input		Transition		Next State
Name	Action	Timer	Event	Action	Condition	Action	
Initial	-	-	X?	-	-	-	State1
State1	-	$\Delta(t_0)$	-	-	C1	Y!	Initial
			-	-	C2	-	Final
Final	-	-	-	-	-	-	-

4.2.2. State Transition Diagram of PM(i,j)

A PM consists of a process part, a chuck, and a slit valve. Therefore, in order to model the operating logic of a PM, the three state graphs for the components are required. In the model, PM(i,j) denotes the j-th parallel PM for the i-th process step.

For the process part, the state graph has six states: 'IDLE', 'PREPARE', 'PROCESS', 'PROCESS_END', 'CLEANING', and 'STOP'. When a wafer is loaded into a PM and a slit valve door is closed, the state of the PM is changed from 'IDLE' to 'PREPARE'. In this study, we assume that a tool follows the earliest starting policy that there are no intentional delays in the tool; hence, the state is changed instantly from 'PREPARE' to 'PROCESS'. If a tool follows another process-start rule, there might be a time delay h_t before entering the process state. The state becomes 'PROCESS_END' after the process time p_t has elapsed. Then the state becomes 'IDLE' after TM unloads the wafer and the slit valve door is closed. A PM cleaning operation can be performed only when the state is 'IDLE'. In order for the cleaning operation to start, the cleaning condition should be satisfied. For a multi-period cleaning operation, the operation starts after the specified number of wafers are processed. For a non-periodic cleaning operation, the operation starts when the degree of contamination inside the PM exceeds the reference value. Such conditions are referred to in the model as (C1). If a cleaning operation starts, the state becomes 'CLEANING'. After a required cleaning time c_t has elapsed, the state returns to 'IDLE'. When a PM receives a '_stop' event message, the state becomes 'STOP' and the simulation is terminated.

The state graph of a chuck has three states: 'UNFIXED', 'FIXED', and 'STOP'. The state graph of a slit valve door also has three states: 'CLOSED', 'OPENED', and 'STOP'. The chuck and the slit valve door work closely together. In order for a wafer to be loaded, the state of the slit valve door should be 'OPENED'. The wafer is then loaded, and the state of the chuck becomes 'FIXED' and the state of the slit valve door becomes 'CLOSED'. Similarly, in order to unload a wafer, the states of the chuck and the slit valve door respectively become 'UNFIXED' and 'OPENED'. After the wafer is unloaded, the state of the slit valve door again becomes 'CLOSED'. When the chuck and the slit valve door receive '_stop' event messages, the states become 'STOP'.

Every operation of the chuck and the slit valve door has its own task time. Therefore, the chuck and the slit valve door send their state transition messages after required task times have passed.

In this study, we propose a model to consider only simplified PM operating logics and PM cleaning operations. However, in actual tools, motion control and wafer alignment check are conducted inside the PM. Furthermore, there are various exceptions such as PM failure and repair, time variations, and wafer alignment failures. An improved model has been developed in order to consider all realistic situations.

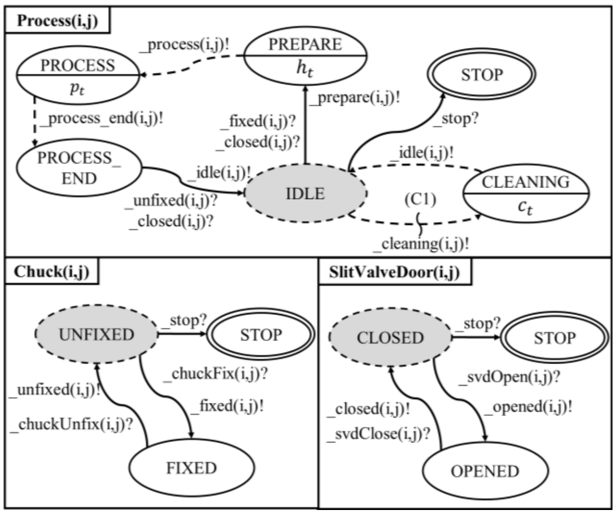

Figure 6: State Transition Diagram of PM(i,j)

Table 3: State Transition Table of PM(i,j)

State		Input	Transition		Next State
Name	Timer	Event	Condition	Action	
IDLE	-	_fixed(i,j)? _closed(i,j)?	-	_prepare(i,j)!	PREPARE
		-	C1	_cleaning(i,j)!	CLEANING
PREPARE	h_t	-	-	_process(i,j)!	PROCESS
PROCESS	p_t	-	-	_process _end(i,j)!	PROCESS _END
PROCESS _END	-	_unfixed(i,j)? _closed(i,j)?	-	_idle(i,j)!	IDLE
CLEANING	c_t	-	-	_idle(i,j)!	IDLE
UNFIXED	-	_chuckFix(i,j)?	-	_fixed(i,j)!	FIXED
FIXED	-	_chuckUnfix(i,j)?	-	_unfixed(i,j)!	UNFIXED
CLOSED	-	_svdOpen(i,j)?	-	_opened(i,j)?	OPENED
OPENED	-	_svdClose(i,j)?	-	_closed(i,j)?	CLOSED

4.2.3. State Transition Diagram of TM

The state transition diagram of a TM is shown in Fig. 7 and Table 4. The state graph model of the TM has three states: 'EMPTY', 'HOLD', and 'STOP'. If the TM holds a wafer, then its state is 'HOLD'. If not, then the state is 'EMPTY'. Each state has a state variable (i,j) that denotes the ID of the PM to which the TM is directed. So that the TM can unload the wafer, the states of the chuck and the slit valve door are 'UNFIXED' and 'OPENED', respectively. After the TM completes its unloading task, it sends an event message '_svdClose' so that the slit valve door can be closed. Likewise, the state of the slit valve door should be 'OPENED' before the TM loads a wafer. After the wafer loading is completed, the TM sends event messages '_chuckFix' and '_svdClose' to the chuck and slit valve door, respectively. When the TM moves to another PM, it updates its state variable (i,j). Finally, the state becomes 'STOP', which is the final state, if the TM receives a '_stop' event message.

Each robot operation has its own task time. Thus, like the chuck and the slit valve door, the TM sends its job completion messages after the simulation time has elapsed the required task time.

In addition to the single-armed robot considered in this study, there exist diverse TMs such as a dual-armed robot, a quad-armed robot, and a robot with independent arms.

Furthermore, TM malfunctions occasionally occur. Future work will address a more comprehensive TM model.

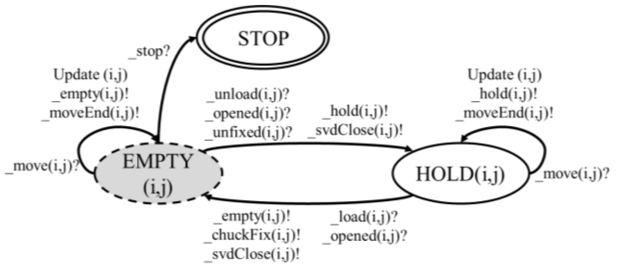

Figure 7: State Transition Diagram of TM

Table 4: State Transition Table of TM

State		Input	Transition		Next State
Name	Timer	Event	Condition	Action	
EMPTY(i,j)	-	_unload(i,j)? _opened(i,j)? _unfixed(i,j)?	-	_hold(i,j)! _svdClose(i,j)!	HOLD(i,j)
		_move(i,j)?	-	Update (i,j) _empty(i,j)! _moveEnd(i,j)!	EMPTY(i,j)
HOLD(i,j)	-	_load(i,j)? _opened(i,j)?	-	_empty(i,j)! _chuckFix(i,j)! _svdClose(i,j)!	EMPTY(i,j)
		_move(i,j)?	-	Update (i,j) _hold(i,j)! _moveEnd(i,j)!	HOLD(i,j)

4.2.4. Tool Scheduler and Module Manager

The tool scheduler determines when each module, especially the TM, should perform what tasks. We apply the open architecture scheduler to the VCT. The tool scheduler issues scheduling commands by referring to the SCFs.

The module manager is responsible for interpreting the scheduling commands into language that PMs and TM can understand and delivering the commands to the modules. In this study, state transitions of the modules occur via event messages; the module manager interprets scheduling commands into several event message sequences and transfers the commands to the modules. For example, when the module manager of the VCT receives a scheduling command 'unload the wafer #1 from PM(1,1)', the module manager sends an event message '_unload(1,1)' to TM and event messages '_svdOpen(1,1)' and '_chuckUnfix(1,1)' to PM(1,1), particularly to SlitValveDoor(1,1) and Chuck(1,1), respectively. If the tool follows other job timing rules, rather than the earliest starting policy, the event messages should include the timing information for each event message.

The module manager is highly related to the tool configuration and the internal operating logic of the modules. The presence of the module manager allows the tool scheduler to operate independently of the tool architecture. Even when two cluster tools have different architectures, it is possible to operate them through a single scheduler when the two module managers are able to interpret the same scheduling commands.

In addition, the module manager informs the scheduler of the states of the modules. The module manager analyzes the state messages sent from the modules and informs the scheduler as to whether each module is abnormal or not.

5. APPLICATION: PERFORMANCE ANALYSIS IN A SINGLE-ARMED CLUSTER TOOL WITH PM CLEANING OPERATIONS

As an application of the proposed framework, we conduct a performance analysis on benchmark schedules in a single-armed cluster tool with PM cleaning operations. The schedules to be analyzed are the backward sequence which guarantees optimal productivity in the basic single-armed cluster tool and the backward(z) sequence which is known to give good performance in a single-armed cluster tool with 1-periodic PM cleaning operations, as proposed by Yu and Lee (2017).

5.1. SCFs for Performance Analysis

For the performance analysis, we first create tool architecture information SCFs. In the analysis, we set all task times of the slit valve door and the chuck 1. TM task times are also set to 3. The number of process steps, the number of parallel PMs, the process times, the cleaning times, and the cleaning periods are set differently according to the experiment.

After this, the benchmark TM task schedule SCFs for the backward and the backward(z) sequences should be created. Both sequences are basically similar in that they are pull-type tool operation methods; however, the initial state of the tool when the tool is operated in the cyclic schedule is different. The conventional backward sequence proceeds at the full loading state while all PMs are in progress and then returns to the full loading state. On the other hand, in order to keep the WIP constantly and increasing the productivity, the backward(z) sequence is used to allocate some of the parallel PMs to PM cleaning operations, where the number of PMs to allocate is denoted by 'z'. This is called a partial loading strategy. Therefore, although both schedules give the same schedule messages, those show a large difference in productivity due to the different operation strategies. For example, consider a tool with six PMs and three process steps. Assume that PM1, PM2 - PM4, and PM5 - PM6 are assigned as parallel PMs for the first, the second, and the third process steps. Table 5 demonstrates the initial states of both sequences when a tool is operated on a cyclic schedule. One way of finding an optimal z value is introduced in Yu and Lee (2017). Fig. 8 shows Petri net models for both schedules. The model has a different net shape and token marking vectors depending on the tool configuration and the cleaning period. The earliest starting rule is applied in the analysis.

In this experiment, we do not consider any exceptions in our analysis of the impact of the cleaning period and time on the performance of each TM schedule.

Table 6: Initial States for Backward and Backward(z) Sequences

Initial State for the Backward Sequence					
PM1(1)	PM2(2)	PM3(2)	PM4(2)	PM5(3)	PM6(3)
Process	Process	Process	Process	Process	Process

Initial State for the Backward(z) Sequence (optimal z = [0, 1, 1])					
PM1(1)	PM2(2)	PM3(2)	PM4(2)	PM5(3)	PM6(3)
Process	Cleaning	Process	Process	Cleaning	Process

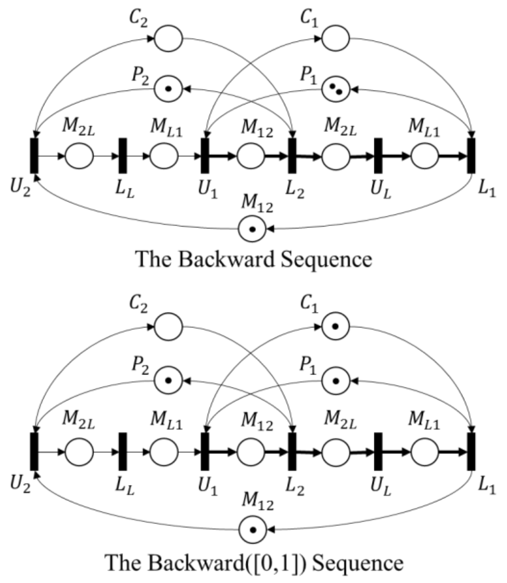

Figure 8: Petri Net Models for the Backward and the Backward(z) Sequences

5.2. Experiment Results

In the experiments, we computed the average cycle times of the backward and backward(z) sequences for the SCFs that we created for diverse tool configuration and both TM schedules.

Problem instances for the experiment and the results are shown in Table 7. In the table, the wafer flow pattern $[m_1, \dots, m_n]$ means that there exist m_i parallel PMs for the i-th process step. This is similar to the process times $[p_1, \dots, p_n]$ and the cleaning times $[c_1, \dots, c_n]$. The cleaning periods are assumed to have the same value for all PMs; this value is larger than one. Since the average cycle time indicates that the average time elapsed to produce a wafer, a lower average cycle time means better performance. The gap shows how the average cycle time of the backward(z) sequence is lower than the average cycle time of the backward sequence. Thus, the negative gap means that the backward sequence yields better performance.

From Table 7, we can see that, for the most cases, the backward(z) sequence provides better performance than the backward sequence. Particularly, the gap becomes larger when the PMs have a short cleaning period and long cleaning times. This indicates that the greater the influence of the PM cleaning processes on the tool operation, the better the productivity of the backward(z) sequence. However, the backward sequence provides better performance for several cases with long cleaning period and short cleaning times. Therefore, in order to increase the productivity, it is important to apply an appropriate schedule according to the tool configuration, the cleaning time, and the cleaning period. Since adaptive scheduling is possible in response to environment changes and time-disruptive exceptions, the suggested open architecture scheduling system for the cluster tool will be competitive.

Table 7: Performance Analysis of the Backward and Backward(z) Sequences.

Tool Configuration				**Average Cycle Time**		
Wafer Flow Pattern	**Process Times**	**Cleaning Times**	**Cleaning Periods**	**Backward**	**Backward(z)**	**Gap (%)**
[4]	[100]	[50]	2	190	150.5	20.8
			3	169.7	141.5	16.6
	[100]	[100]	2	199.5	168.8	15.4
			3	178.6	161.33	9.7
[1, 3]	[70, 300]	[35, 150]	2	152.8	164.5	-7.7
			3	129.1	164.5	-27.4
		[70, 300]	2	221.2	181.5	17.9
			3	188.4	171.8	8.8
[1, 4]	[70, 400]	[35, 200]	2	168	148.4	11.7
			3	150.7	148.4	1.5
		[70, 400]	2	275.6	154.9	43.8
			3	215.5	147.3	31.6
[2, 3]	[150, 300]	[75, 150]	2	298	268.5	9.9
			3	278	238.7	14.1
		[150, 300]	2	459.1	279.5	39.1
			3	385	250.8	34.9
[1, 3, 2]	[50, 300, 150]	[25, 150, 75]	2	158.3	164.5	-3.9
			3	144.7	164.5	-13.7
		[50, 300, 150]	2	241.4	191.2	20.8
			3	203	182	10.3

6. CONCLUSION

We have suggested a framework of an open architecture scheduling system for a single-armed cluster tool with PM cleaning operations. In the system, the tool scheduler refers to SCFs and performs scheduling. SCFs can be freely replaced from outside. To apply the framework, we have developed a VCT similar to the actual cluster tool. The scheduler, the module manager, the PMs, and the TM are modeled to describe the VCT. Using the VCT, we have also conducted performance analysis on the backward and backward(z) sequences for a single-armed cluster tool with multi-period PM cleaning operations. As a result, we found that the open architecture scheduling system that can modify or change the tool schedules based on various tool environments is effective at increasing productivity.

In future works, we will specify SCFs to enable scheduling of various tool structures and operational constraints. Furthermore, exception-handling techniques for various exceptions will be examined.

ACKNOWLEDGMENTS

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2015R1D1A1A01057131).

REFERENCES

Choi B.-K. and Kang D., 2013. Modeling and Simulation of Discrete Event Systems. John Wiley & Sons.

Geismar H. N., Dawande M., and Sriskandarajah C., 2004. Robotic cells with parallel machines: Throughput maximization in constant travel-time cells. Journal of Scheduling, 7 (5), 375–395.

Joo Y.-J. and Lee T.-E., 2004. A Virtual Cluster Tool for Testing and Verifying a Cluster Tool Controller and a Scheduler. IEEE Robotics & Automation Magazine, 11 (3), 33-49.

Kim C. and Lee T.-E., 2013. Modelling and simulation of automated manufacturing systems for evaluation of complex schedules. International Journal of Production Research, 51 (12), 3734-3747.

Kim H., Kim H.-J., Lee J.-H., and Lee T.-E., 2013. Scheduling dual-armed cluster tools with cleaning processes. International Journal of Production Research, 51 (12), 3671–3687.

Kim H.-J., Lee J.-H., and Lee T.-E., 2015. Noncyclic scheduling of cluster tools with a branch and bound algorithm. IEEE Transactions on Automation Science and Engineering, 12 (2), 690–700.

Kim H.-J., Lee J.-H., Jung C., and Lee T.-E., 2013. Scheduling cluster tools with ready time constraints for consecutive small lots. IEEE Transactions on Automation Science and Engineering, 10 (1), 145–159.

Kim J.-H., Lee T.-E., Lee H.-Y., and Park D.-B., 2003. Scheduling analysis of time-constrained dual-armed cluster tools. IEEE Transactions on Semiconductor Manufacturing, 16 (3), 521–534.

Lee J.-H., Kim H.-J., and Lee T.-E., 2014. Scheduling cluster tools for concurrent processing of two wafer types. IEEE Transactions on Automation Science and Engineering, 11 (2), 525–536.

Lee, J.-H., and Lee, T.-E., 2010. An open scheduling architecture for cluster tools. In Proceedings of the 6th IEEE Conference on Automation Science and Engineering, pp. 420-425, Toronto (Ontario, Canada).

Lee, T.-E., 2008. A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In Proceedings of the 40th Conference on Winter Simulation, pp. 2127-2135, Miami (Florida, USA).

Min Y. and Lin X.-R., 2013. A Simulation-Based Analysis of Cluster Tools Scheduling with Plant Simulation. In Proceedings of the 19th International Conference on Industrial Engineering and Engineering Management, pp. 71-80. Berlin (Germany),

Niedermayer H. and Rose O., 2003. A simulation-based analysis of the cycle time of cluster tools in semiconductor manufacturing. In Proceedings of the 15th European Simulation Symposium, pp. 349-354, Delft (Netherlands).

Pan C., and Bao N., 2012. An eM-Plant-based virtual single-arm cluster tool. In Proceedings of the 9th IEEE International Conference on In Networking, Sensing and Control, pp. 40-45, Beijing (China).

Rostami S., Hamidzadeh B., and Camporese D., 2001. An optimal periodic scheduler for dual-arm robots in cluster tools with residency constraints. IEEE Transactions on Robotics and Automation, 17 (5), 609–618.

Shin Y.-H., Lee T.-E., Kim J.-H., and Lee H.-Y., 2001. Modeling and implementing a real-time scheduler for dual-armed cluster tools. Computers in Industry, 45, 13-27.

Yu T.-S., Kim H.-J., and Lee T.-E., 2017, Scheduling single-armed cluster tools with chamber cleaning operations. IEEE Transactions on Automation Science and Engineering, DOI: 10.1109/TASE.2017.2682271.

Venkatesh S., Davenport R., Foxhoven P., and Nulman J., 1997. A steady-state throughput analysis of cluster tools: Dual-blade versus single-blade robots. IEEE Transactions on Semiconductor Manufacturing, 10 (4), 418–424.

Wu N. Q., Chu F., Chu C., and Zhou M. C., 2011. Petri net-based scheduling of single-arm cluster tools with reentrant atomic layer deposition processes. IEEE Transactions on Automation Science and Engineering, 8 (1), 42–55.

Wu N. Q., Chu C., Chu F., and Zhou M. C., 2008. A petri net method for schedulability and scheduling problems in single-arm cluster tools with wafer residency time constraints. IEEE Transactions on Semiconductor Manufacturing, 21 (2), 224–237.