Scheduling Single-Armed Cluster Tools With Reentrant Wafer Flows

Hwan-Yong Lee and Tae-Eog Lee

***Abstract*—A cluster tool for semiconductor manufacturing consists of several single-wafer processing chambers and a wafer-handling robot in a closed environment. The use of cluster tools is extended to reentrant processes such as atomic layer deposition, where a wafer visits a processing chamber more than once. Such a reentrant wafer flow complicates scheduling and control of the cluster tool and often causes deadlocks. We examine the scheduling problem for a single-armed cluster tool with various reentrant wafer flows. We develop a convenient method of modeling tool operational behavior with reentrant wafer flows using Petri nets. By examining the net model, we then develop a necessary and sufficient condition for preventing a deadlock. We also show that the cycle time for the asymmetric choice Petri net model for a reentrant wafer flow can be easily computed by using the equivalent event graph model. From the results, we systematically develop a mixed integer programming model for determining the optimal tool operation sequence, schedule, and cycle time. We also extend a workload measure for cluster tools with reentrant wafer flows. Finally, we discuss how our results can be used for engineering a cluster tool. We compare two proposed strategies, sharing and dedicating, of operating the parallel processing chambers for identical process steps.**

***Index Terms*—Cluster tools, deadlock, Petri net, reentrant wafer flow, scheduling.**

I. Introduction

THE semiconductor manufacturing industry has developed new wafer fabrication technologies including single-wafer processing, rapid thermal processing, mini-environment, and standard mechanical interface (SMIF). Due to the technologies, several single-wafer processing chambers and a wafer-handling robot are integrated into a closed mini-environment. Such an integrated tool, called a *cluster tool*, may have diverse architectures, depending on the chamber configuration, radial or linear, the number of robot arms, single- or dual-armed, presence of internal buffer modules, and so on. Fig. 1 illustrates a single-armed radial-type cluster tool with four processing chambers. Although cluster tools are apparently simple, they have many scheduling issues. There are works on scheduling single- or dual-armed cluster tools [1]–[9] and similar hoist scheduling problems [10], [11].

Manuscript received November 22, 2004; revised December 5, 2005. This work was supported by the Brain Korea 21 Fund for 2003 and 2004.

H. Y. Lee is with the High Definition Display Center LCD Business, Samsung Electronics Company, Ltd., Chungcheongnam-do 336-841, Korea.

T.-E. Lee is with the Department of Industrial Engineering, Korea Advanced Institute of Science and Technology (KAIST), Taejon 305-701, Korea, and also with the Decision and Control Laboratory, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801 USA (e-mail: telee@kaist.ac.kr).

Digital Object Identifier 10.1109/TSM.2006.873402

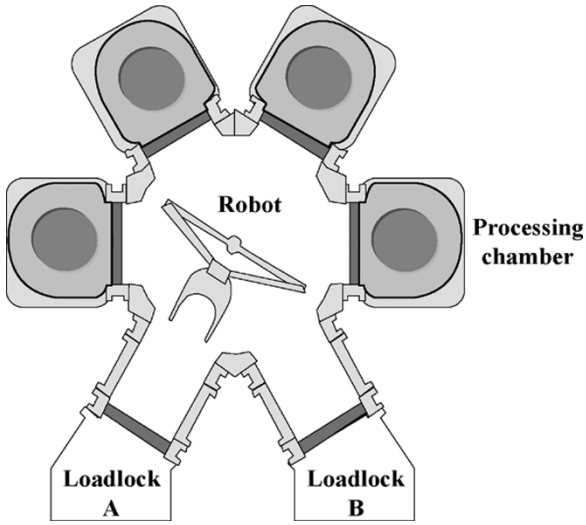


Fig. 1. Single-armed cluster tool with four chambers.

Cluster tools have been increasingly used for most fabrication processes including lithography, etching, deposition, and even inspection. Atomic layer deposition (ALD) technology that makes film deposition with mono-atomic layer precision is increasingly used for deposition processes. As compared with conventional chemical–vapor deposition (CVD), it can easily control particle generation and uses a significantly lower processing temperature (100 °C $\sim$ 400 °C instead of 550 °C $\sim$ 800 °C). ALD easily controls the film thickness by repeating the deposition cycle as needed, while CVD determines the thickness by adjusting the deposition time. Therefore, a wafer for ALD takes a sequence of identical process steps more than once. Some other processes, even CVD, also sometimes repeat identical process steps. Such identical process steps are often performed at a single chamber, and hence the chamber is visited by a wafer as many as the number of identical process steps that share the chamber. There are several reasons. First, as in ALD processes, the repeated process steps often require exactly identical processing conditions for each repetition cycle. Therefore, it is desirable to perform the repeated process steps for a wafer at a single chamber, not by additional processing chambers. Otherwise, it may be hard to control the processing conditions. Discrepancy between the processing conditions of different cycles may be detrimental for wafer quality and uniformity.

Second, it is costly or even impossible to add a processing chamber due to space or footprint restrictions. Therefore, it is more economical to perform identical process steps at a single chamber. A wafer flow that visits a single processing chamber

more than once is called *reentrant*. Such a chamber and the repeated process steps are also called *reentrant*. Sharing a chamber by multiple process steps complicates tool scheduling significantly and even causes a deadlock. Therefore, tool vendors tend to use simple reentrant wafer flow patterns, although they need more complicated reentrant wafer flows in order to fully utilize modern process technologies such as ALD.

While reentrant wafer flows are increasingly used for cluster tools, there have been only two works on scheduling cluster tools with reentrant wafer flows [5], [12]. Perkinson *et al.* [3] provide only a rough bound on the cycle time for a single-armed tool with reentrant wafer flows. Zuberek [5] suggests a free-choice Petri net model for a dual-armed cluster tool with a reentrant wafer flow when a particular deadlock-free robot task sequence is assumed. He claims that since the net model is a free-choice Petri net, which still needs token routing decisions at some places, the cycle time should be computed by the invariant analysis, more complicated analysis, than critical circuit analysis for an event graph model. However, Jeng [18] indicates that the free-choice net model does not need token routing decisions due to its special structure and hence is behaviorally equivalent to an event graph, for which the cycle time is easily computed as the critical circuit ratio. Zuberek [5] does not address how a deadlock can be prevented and the optimal robot task sequence should be determined. For the same tool model, Kim [12] proposes two heuristic scheduling strategies for preventing deadlocks by modifying the prevalently swap operation sequence, which causes deadlocks for reentrant wafer flows. However, it is not clear yet whether or how the performance advantages of dual arms can be fully utilized for reentrant wafer flows. A single-armed robot can hold only one wafer at a time. Therefore, it is more difficult to schedule a single-armed cluster tool with reentrant wafer flows. Some studies on hoist-based production systems, which are operationally similar to single-armed cluster tools, deal with reentrant job flows [10], [11]. However, they assume limited reentrance models that allow reentrance only at the first or the last process steps or do not allow multiple reentrances and parallel machines for a process step. They focus on developing optimization algorithms rather than Petri net modeling and deadlock analysis. For cluster tools with various configurations and realistic reentrant wafer flow patterns, systematic ways of modeling the tool behavior and finding an optimal deadlock-free robot task sequence are yet to be examined.

Dual-armed cluster tools tend to be used for reducing the cycle time for cluster tools with a higher robot task workload. Dual-armed cluster tools mostly perform robot tasks by performing a sequence of swap operations for the PMs, each of which unloads a wafer from a chamber onto an empty robot arm and loads the next wafer on the other arm into the chamber. Such a swap operation simplifies the robot task scheduling and is known to optimize the cycle time. However, when there is a reentrant wafer flow, the robot task scheduling-based swap operations cause a deadlock [12]. Therefore, the robot task scheduling becomes complicated and the merit of dual arms may diminish. That is, the two arms may not be fully utilized. While dual-armed robots are increasingly used for shorter tool cycle time, single-armed robots are still widely used due to lower cost, improved robot speed, and even simplified scheduling. Even a dual-armed tool, when one arm is malfunctioning or contaminated, performs single-arm operation by using another arm.

Most single-armed cluster tools without reentrant wafer flows popularly use the *backward sequence* that first unloads the wafer from a chamber i and loads the wafer unloaded from chamber $i-1$ into chamber i. It is shown that the backward sequence is optimal when the workloads of the chambers are reasonably balanced [7]. However, when the wafer visits a chamber more than once, the processing order between the reentrant wafers should be determined. Therefore, the choice of the processing order, in conjunction with the robot task sequence, may cause a *circular wait* between a pair of chambers, each holding a wafer, where the robot tries to unload a wafer from one chamber and load it into the other chamber. This causes a deadlock from which the tool cannot proceed. Therefore, we should develop a new way of determining the robot task sequence.

In this paper, we examine the scheduling problem for a single-armed radial-type cluster tool with various reentrant wafer flows. We develop a convenient method of modeling tool operational behavior with reentrant wafer flows using Petri nets. By examining the net model, we then develop a necessary and sufficient condition for preventing a deadlock. We also show that the cycle time for the asymmetric choice Petri net model for a reentrant wafer flow can be easily computed by using the equivalent event graph model. From the results, we systematically develop a mixed integer programming model for determining the optimal tool operation sequence, schedule, and cycle time. We also extend a workload measure for cluster tools with reentrant wafer flows. Finally, we discuss how our results can be used for engineering a cluster tool. We compare the two proposed strategies, sharing and dedicating, of operating the parallel processing chambers for identical process steps.

II. Petri Net Modeling for Reentrant Wafer Flows

Petri nets are popularly used for modeling and analyzing operational behavior of automated systems. A Petri net is a graphical and mathematical framework for modeling and analyzing discrete-event systems [13]. Bars, circles, arrows, and dots indicate transitions, places, arcs, and tokens, respectively. They usually model activities or events, conditions or activities, precedence between transitions and places, and entities or conditions, respectively. A token at a place means that the corresponding condition is satisfied or the corresponding activity is in progress. Time delays for modeling activity times are associated with transitions and/or places. When a place models an activity, a token at the place can join enabling the succeeding transitions after its sojourn time that is the specified time required for the corresponding activity. A transition is enabled when each input place has a token and the sojourn time of a token at each timed input place has elapsed. It is known that a Petri net model with time delays at transitions is equivalently transformed into one with time delays at places and vice versa. An enabled transition fires after its firing delay, if any. After a transition fires, a token is removed from each input place and a token is added to each output place. A detailed formal presentation of Petri nets can be found in [13]. There have been works on Petri net modeling of cluster tools [4]–[9].

We briefly explain why a Petri net-based approach is useful for cluster tool scheduling. A Petri net model clarifies relationships between the events and activities in cluster tool operation. Precedence relations between the events and synchronization of the activities are clearly modeled by transitions and places. The locations of the tokens at the places and their movements explain the state of the tool and the dynamic state changes, respectively. Therefore, by examining the Petri net model, we can identify why a deadlock occurs and how long the cycle time is. A cluster tool is often operated to repeat a predefined work cycle such that each PM and the robot performs a predefined activity or task sequence cyclically. Such a cyclic scheduling method has advantages such as predictable behavior, easier mathematical analysis, better cycle time optimization, and even constant wafer delays since a steady schedule with identical timing patterns for each cycle can be computed due to cyclic operation [8], [14]–[16]. A cluster tool, when the robot task sequence is determined, repeats such a cyclic work cycle. Therefore, the tool's operational behavior is modeled by a special class of Petri nets, called a decision-free event graph, for which not only the cycle time but also steady earliest schedule patterns, most desirable schedule patterns, can be systematically computed [14]–[16]. Furthermore, from the precedence relations between the events of the event graph model, we can easily derive a linear programming model that determines the cycle time and a steady schedule. Therefore, by identifying how the robot task sequence decision affects the precedence relations in the event graph model, we can extend the linear program into a mixed integer program that determines the optimal robot task sequence as well as the cycle time and an optimal steady schedule. Consequently, it is desirable to first develop an event graph model for a cluster tool with a given robot task sequence and then derive a mixed integer program model from the linear program associated with the event graph model to optimize the robot task sequence.

We now explain a systematic Petri net modeling method for a single-armed cluster tool with reentrant wafer flows. To do this, we develop Petri net submodels for reentrant wafer flows, the robot work cycle, and the PM work cycle and then combine them into the whole model and develop the initial marking rule that assigns the tokens to the places.

A. Modeling Reentrant Wafer Flows

When the last wafer is unloaded from a loadlock, the next wafer is unloaded from the other loadlock. Due to such pipelined operation, the tool repeats steady work cycles until all lots of identical wafers are completed. The process time for each process step and the robot task times (unloading, loading, and transporting times) are known. A processing chamber is also called a *processing module* (PM). We also assume that there is no disruptional event such as breakdowns of the PMs or the robot. The process steps that a wafer undergoes are sequentially numbered from 1 to n. The operation for returning a finished wafer into the loadlock and unloading a new wafer from the loadlock is numbered $n+1$. A wafer flow pattern is then defined as $(m_1^{k_1}, m_2^{k_2}, \ldots, m_n^{k_n})$, where m_i indicates the number of PMs available for process step i, and k_i is the serial index for a reentrant PM where process step i is performed. The reentrant PMs are sequentially indexed from one. $k_i = j$ implies that the ith process step is performed by the jth reentrant PM. A nonreentrant PM i has index $k_i = 0$ and we omit such an index 0. For instance, $(1^1, 1^2, 2, 1^1, 1^2)$ indicates wafer flow pattern ($\mathrm{PM1} \rightarrow \mathrm{PM2} \rightarrow \mathrm{PM3}$ or $\mathrm{PM4} \rightarrow \mathrm{PM1} \rightarrow \mathrm{PM2}$), where identical process steps 1 and 4 and identical process steps 2 and 5 are performed by PM1 and PM2, respectively. The loadlock step $n+1$ is omitted in the wafer flow pattern notation.

The position on the robot arm where a wafer unloaded from a loadlock locates is adjusted at the aligner near the loadlock for accurate positioning. Since the aligning time is relatively short, within a few seconds, it is included in the time for unloading a wafer from a loadlock. For each process step $i = 1, \ldots, n+1$, the timed transitions that represent unloading a wafer from a module (a PM or a loadlock) and loading the wafer into the module are denoted by U_i and L_i, respectively. T_i represents a timed place for transporting a wafer from a PM to the PM for the next process step. Let P_i indicate the timed place for processing a wafer at a PM. Then, the subgraph $U_1 \rightarrow T_1 \rightarrow L_1 \rightarrow P_1 \rightarrow \cdots \rightarrow U_n \rightarrow T_n \rightarrow L_n \rightarrow P_n \rightarrow U_{n+1} \rightarrow T_{n+1} \rightarrow L_{n+1}$ represents the sequence of operations that a wafer undergoes. Once the robot loads a wafer into a PM, it may wait at the PM until it unloads the wafer after the PM completes processing the wafer or moves to another PM. Such robot waiting is modeled by place W_i from transition L_i to transition U_{i+1} for each PM i. Place M_{ij} represents for the robot move with the arm empty from a PM for process step i to the PM for process step j. Even if a robot's waiting place W_i may look redundant to processing place P_i, we keep the waiting places in order to model the robot's state explicitly.

We note that for modeling convenience, we let both transitions and places be timed. A transition usually indicates an event or activity. When it indicates an event, it has no firing delay, that is, it fires as soon as it is enabled. However, when it models an activity, it has a firing delay, that is, it fires after the firing delay since it is enabled. A place can represent a condition without any associated time or models an activity by associating a token holding time as much as the activity duration with the place. In the latter case, each token that enters such a timed place can contribute to enabling the output transitions after the token holding time elapses. In our Petri net models, all transitions are timed because they indicate robot tasks. All places except A_i and W_i are also timed, since they indicate robot tasks or processing activities at PMs. Of course, each timed transition can be modeled by a sequence of an untimed transition, a timed place, and an untimed transition, which indicates the start of an activity, the activity, and the completion of the activity, respectively. A timed place also can be similarly modeled by a sequence of an untimed place, a timed transition, and an untimed place. However, our modeling strategy of allowing both transitions and places to be timed makes the model more compact.

B. Modeling Robot Work Cycle

An unloaded wafer should be loaded into a PM before the robot becomes free and can perform some other task. The wafer cannot be returned into the loadlock before it completes all process steps because the wafer in progress, which is still hot,

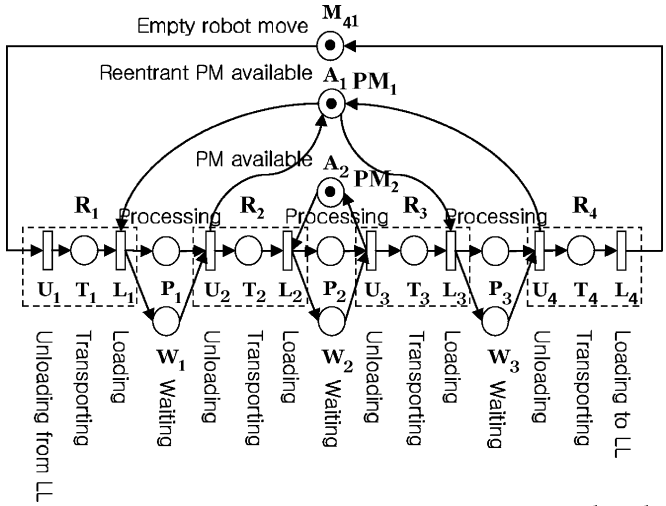


Fig. 2. Petri net model for $\sigma^1 = (1, 2, 3, 4)$ of wafer flow pattern $(1^1, 1, 1^1)$.

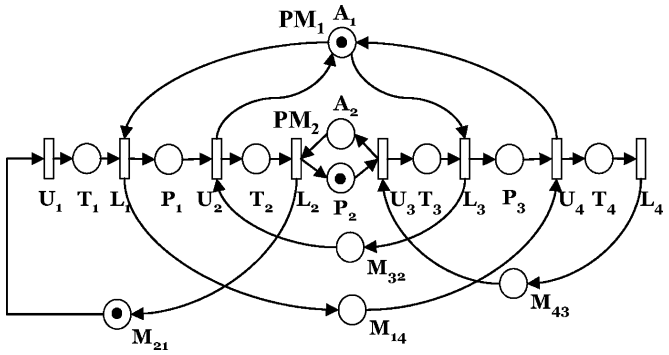


Fig. 3. Petri net model for $\sigma^2 = (1, 4, 3, 2)$ of wafer flow pattern $(1^1, 1, 1^1)$.

should not affect the other wafers in the loadlock and should not be cooled down too much before it takes the next process step. The wafer cannot be stored temporarily at any other PM without taking processing there. It is because a PM starts processing as soon as a wafer is loaded into the PM in order to avoid the unnecessary side effects due to the residual gases and heat within the PM. Therefore, once a wafer is unloaded from a PM for process step $i-1$, it should be transported to a PM for the next process step i and loaded into the PM. Therefore, the robot tasks of performing these operations, U_i, T_i, and L_i, should be performed in sequence. The subsequence of the robot tasks $U_i \rightarrow T_i \rightarrow L_i$ cannot be changed. The robot tasks can be regarded as an aggregated task R_i for preparing for process step i. Consequently, the robot task sequencing problem is reduced to a sequencing problem for the aggregated robot tasks R_is. They are marked by dotted rectangles in Fig. 2.

The robot work cycle is then modeled by a circuit that connects all transitions for robot tasks and the associated places U_i, L_i, T_i, M_{ij}, and W_i. The connecting rules follow. First, each place for a transporting task T_i should connect from transition U_i to L_i. Second, an empty moving task M_{ij} should link from transition L_i to U_j. Therefore, the subgraphs R_i, $i = 1, \ldots, n+1$ should be connected altogether to form a circuit called a *robot circuit* that represents a robot work cycle. The cyclic order in which the subgraphs are connected defines the sequence in which the robot tasks are performed. Such a robot task sequence is represented by function $\sigma \equiv (\sigma(1), \ldots, \sigma(n))$, where $\sigma(i)$ indicates the ith aggregated robot task. For instance, $\sigma = (1, 2, \ldots, n)$ corresponds to a robot circuit, $R_1 \rightarrow W_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_n \rightarrow W_n \rightarrow R_{n+1} \rightarrow M_{(n+1)1} \rightarrow R_1$. We observe that the robot task sequence σ determines the tool operation sequence. The robot circuit should have one token that indicates availability or the state of the robot. We let the moving place prior to transition U_1 have the token at the initial state. This implies that the robot is moving to the loadlock or ready for unloading a new wafer from the loadlock.

C. Modeling PM Work Cycles

Since each process step is repetitively performed by the assigned single or parallel PMs, it has a circuit called a *processing circuit*, $(A_i \rightarrow L_i \rightarrow P_i \rightarrow U_{i+1} \rightarrow A_i)$. A processing circuit should have tokens as many as the number of parallel chambers for the process step. The tokens in a processing circuit should be appropriately located at the places in the circuit, processing places P_i or PM availability places A_i. The identical process steps may share an identical PM. For instance, process steps 1 and 3 for wafer flow pattern $(1^1, 1, 1^1)$ share PM1. Therefore, the processing circuits for the process steps at the same PM share the PM availability place A_1. We let the availability place have only one token. By combining the robot circuit and the processing circuits, the Petri net model for a given robot task sequence of a wafer flow pattern is determined. Fig. 2 illustrates the Petri net model for sequence $\sigma = (1, 2, 3, 4)$ of wafer flow pattern $(1^1, 1, 1^1)$. Since identical process steps 1 and 3 are processed by PM1, a token at place A_1 for availability of the reentrant PM is shared by the two processing circuits $L_1 \rightarrow P_1 \rightarrow U_2 \rightarrow A_1$ and $L_3 \rightarrow P_3 \rightarrow U_4 \rightarrow A_1$. Fig. 3 illustrates the Petri net model for a different sequence $\sigma = (1, 4, 3, 2)$ for the same wafer flow pattern. In the figure, place M_{14} represents a robot move from PM1 to PM3 after loading at PM1 (transition L_1), where place P_3 indicates the second processing of a wafer at PM1. For other robot task sequences, the Petri net models can be similarly configured.

We note that our modeling method defines a processing circuit for each process step and makes the processing circuits of the process steps performed at an identical reentrant PM share the availability place of the PM. Such a modeling strategy clarifies the robot work cycle as well as the wafer flow pattern. ALD processes may have reentrant wafer flow patterns more complicated than the illustrated examples. A wafer may visit a PM more than two times. There may be more than one reentrant PM. Our proposed modeling method can easily model such general reentrant flow patterns. Figs. 4 and 5 illustrate the Petri net models for wafer flow patterns $(1^1, 1, 1^1, 1, 1^1)$ and $(1^1, 1^2, 1^1, 1^2)$, respectively.

D. Modeling Operating Strategies for Parallel PMs for Identical Process Steps

We now consider the case where the identical process steps share more than one PM. The workload of a reentrant PM tends to increase as the number of reentrances to the PM becomes larger. Therefore, in order to reduce the workload at a reentrant PM, we may add PMs for the identical process steps. There can be different strategies of operating such multiple PMs $M \equiv$

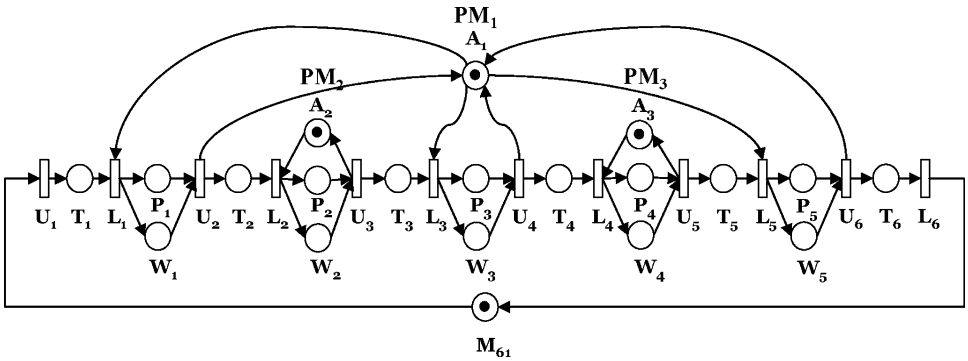


Fig. 4. Petri net model for wafer flow pattern $(1^1, 1, 1^1, 1, 1^1)$.

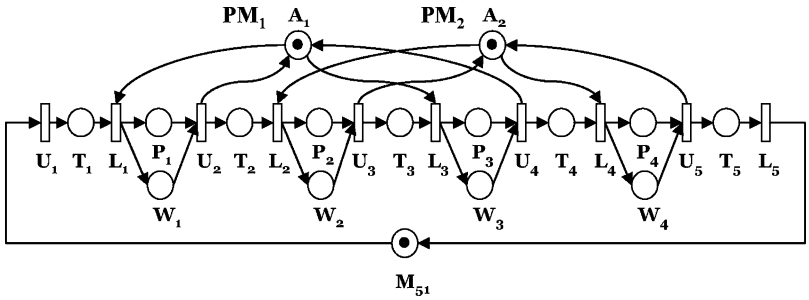


Fig. 5. Petri net model for wafer flow pattern $(1^1, 1^2, 1^1, 1^2)$.

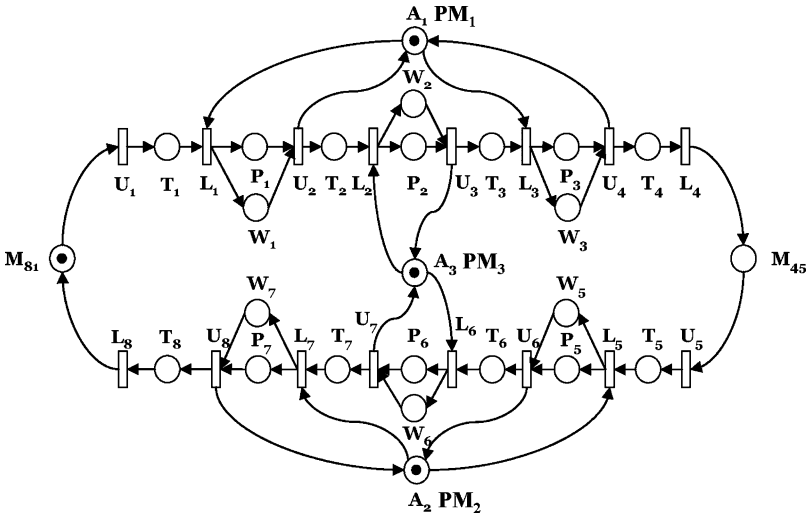


Fig. 6. Petri net model for sharing strategy for wafer flow pattern $(2^1, 1, 2^1)$.

$\{PM_{k_1}, PM_{k_2}, \ldots, PM_{k_l}\}$ assigned for the identical reentrant process steps $\Gamma \equiv \{i_1, i_2, \ldots, i_r\}$. When the number of PMs l is equal to the number of identical process steps r, there are two different strategies of using the PMs. First, the *dedicating strategy* is to assign each PM of M to each process step of Γ. In this case, the wafer never visits the same PM again. Therefore, there is no reentrance and no PM is shared by more than one process step. It is simple to develop the Petri net model for the dedicating strategy. It suffices to assign the availability place for each PM of M to each processing circuit for process steps of Γ. Those availability places are not shared by the processing circuits for Γ. Second, the *sharing strategy* is to allow the process steps of Γ to share the PMs of M. A possible way of sharing is to perform all process steps of Γ at a PM of M. That is, a wafer visits a PM of M as many as the number of identical process steps r. Therefore, the processing circuits for the process steps of Γ share the availability place of the PM. Therefore, the PMs of M are cyclically used by the wafers for fair use. This operation method is required when the processing conditions for the process steps of M should be strictly identical for each wafer. If we let the process steps of Γ use the PMs of M cyclically, the case reduces to the dedicating strategy. Fig. 6 illustrates a Petri net model for the sharing strategy for flow pattern $(2^1, 1, 2^1)$, where a wafer visits only the PM that it visited first. We have $M = \{\mathrm{PM1}, \mathrm{PM2}\}$ and $\Gamma = \{1, 3\}$. Linear subgraphs from U_1 to L_4 and from U_5 to L_8 model the flows of the wafers that visit

the first reentrant PM $\mathrm{PM1}$ and the second one $\mathrm{PM2}$, respectively. Places A_1 and A_2 represent the availability of the first and second reentrant PMs (PM1 and PM2), respectively. The token at place A_1 (or A_2) is shared by the processing circuits for the identical process steps 1 and 3 of the wafer flow that visits the reentrant PM, PM1 (or PM2). Place A_3 indicates the availability of PM3 for process step 2, nonreentrant. PM3 is shared by the two separated wafer flows, not by the identical process steps of each wafer. Places P_2 and P_6 indicate processing at PM3 for the two separated wafer flows for process step 2.

When $l \neq r$, there are more complications. When $l < r$, it is not possible to dedicate all PMs of M to the process steps of Γ. Some process steps of Γ should share some PMs of M. When $l > r$, each process step can be performed by a dedicated PM of M, and $l - r$ surplus PMs are used as parallel PMs for a process step. This case has no reentrance. The sharing strategy can be used for the cases of $l < r$ or $l > r$. While there can be many different ways of assigning the PMs to the process steps of each specific wafer, a fair sharing method is to assign the PMs of M in a cyclic order each time a wafer requests a PM for a process step of Γ. Regardless of the detailed of assigning the PMs, we can easily model the tool behavior by putting tokens at the availability place for the PMs of M as many as l and letting all processing circuits for process steps of Γ to share the tokens at the availability place.

E. Assigning Tokens

Even though we explained the marking of the Petri net models, we further discuss the marking of the tokens within each processing circuit. The parallel PMs for a process step are modeled by as many tokens in the corresponding processing circuit. We do not specify the order in which the parallel PMs are used. They may be used in a cyclic order or in any other sequence. When we wish to specify a specific cyclic order of using the parallel PMs, the net should be appropriately modified [8]. The number of tokens at place P_i for a nonreentrant process step i represents the number of wafers in progress at the process step. More tokens at such a processing place lead to a shorter cycle time at the process step and hence result in a higher or at least the same wafer throughput rate. Then, we may put m_i tokens at the processing place P_i for each nonreentrant process step i. However, this may cause an absurd operation such that the robot tries to load a wafer into a PM when no PM is available. The robot may erroneously try to unload a wafer from an empty PM. We therefore need an appropriate initial marking rule.

F. Initial Marking Rule

1) The robot circuit should have only one token for availability of the robot arm. Without loss of generality, we put the token at the input place M_{j1} to transition U_1 for unloading a new wafer from the loadlock. Therefore, at least one PM for the first process step should be available to load the wafer. Hence, at least one token of the processing circuit for the first process step should locate at the corresponding PM availability place of the processing circuit A_1.
2) The processing circuit for process step i should have m_i tokens, which are at availability place A_i or processing place P_i. When the robot token flows along the robot circuit and reaches transitions L_i or U_i, their input places A_i or P_i, respectively, should have at least one token of the corresponding processing circuit of process step i. That is, for each pair of places (P_i, A_i), if the associated loading transition L_i appears prior to the associated unloading transition U_{i+1} in the robot circuit, A_i should have a token; otherwise, P_i should have a token. Otherwise, the robot tries to load a wafer into an occupied chamber or unload a wafer from the empty chamber, respectively. As far as the mentioned condition is satisfied, we put tokens at processing place P_i as many as possible, but no more than m_i, in order to maximize the throughput rate.

We note that the initial marking rule ensures that the Petri net model does not have any null circuit. It is known that such a null circuit causes a deadlock in an event graph, a special class of Petri nets such that each place has only one input and output transitions. It is known that when a single-armed cluster tool has no reentrant wafer flows, the Petri net model for a given robot task sequence is an event graph, and nonexistence of a null circuit ensures freedom from deadlocks. However, our Petri net models for the tool model with reentrant wafer flows are not event graphs due to the shared availability places A_i for reentrant PMs. Deadlock freedom of the Petri net models should be further examined.

III. Deadlock Analysis

We examine the Petri net models for single-armed cluster tools with reentrant wafer flows and characterize the deadlock-free condition. For a single-armed cluster tool with no reentrant wafer flow, the backward sequence $\sigma = (1, n, n-1, \ldots, 2)$ performs the robot tasks in the backward order of the process steps except the first loadlock. That is, after unloading a wafer from the loadlock and loading it into a PM for the first process step, the robot unloads a wafer from a PM of the last process step n into the loadlock, moves to a PM of process step $n-1$, unloads a wafer from the PM, and loads it into a PM of process step n. The robot then moves to a PM of process step $n-2$ and transfers a wafer from the PM to a PM of process step $n-1$. Such wafer transfer from the preceding process step to the next process step is repeated. For a single-armed cluster tool with no wafer reentrance, the backward sequence is known to be efficient and simplify the tool behavior. Therefore, most single-armed cluster tool vendors use the backward sequence. Lee *et al.* [7] show that the backward sequence minimizes the cycle time when the workloads for the process steps are balanced within some range. However, the backward sequence causes a deadlock for the case of reentrant wafer flows. To apply the backward sequence, at least one PM for each process step should have a wafer before a wafer is unloaded from the last process step and returned into a loadlock. Fig. 3 illustrates the backward sequence $\sigma^2 = (1, 4, 3, 2)$ applied for a reentrant wafer flow pattern $(1^1, 1, 1^1)$. The robot loads a new wafer from the loadlock into PM1 for process step 1. Then, in order to transfer a wafer from PM2 for process step 2 to PM1 for process step 3, the robot should unload from PM1 a wafer that completed process step 3 and return it

into the loadlock. However, PM1 already has a wafer for process step 1. Therefore, this sequence is infeasible. Such an infeasible situation corresponds to a deadlock in terms of the Petri net behavior where no further firing of some transition can be made. It can be easily verified that any different initial marking for the case cannot make the backward sequence feasible. It is because the robot tries to unload a wafer from PM1 that is expected to complete a specific process step while the PM is being occupied by a wafer that took a different process step there. A way of resolving this problem is to let the robot wait at PM1 after loading a wafer into the PM and unload the wafer when the PM completes processing the loaded wafer. That is, the robot should not leave the reentrant PM until the robot unloads the loaded wafer and makes the PM free. We call such scheduling policy *load-and-wait policy*. While for such a simple wafer flow pattern, it is easily seen that the load-and-wait policy does not cause a deadlock, and the effectiveness of the policy for general wafer flow patterns should be further examined.

We now develop a deadlock-free condition for general reentrant flow patterns. To do this, we examine the robot task sequences. A single-armed cluster tool has two types of deadlock situations. First, the robot tries to load a wafer at a PM that is already occupied by another wafer. This situation is called *type I deadlock*. Second, the robot tries to unload a wafer that completed a specific process step from a PM; however, the PM has no such wafer. The PM has no wafer or has only a wafer that completed a different process step. Therefore, the robot cannot proceed the tasks. This situation is called *type II deadlock*. A type II deadlock occurs at a reentrant PM that is shared by more than one process step. There are two causes of deadlocks, an improper marking and an invalid robot task sequence. Improper markings can be prevented by a proper initial marking rule when the robot task sequence is appropriately determined. Therefore, we focus on the robot task sequence. For aggregated robot tasks, R_1 and R_2, we let $R_1 \ll R_2$ denote that R_1 precedes R_2. For a process step i, the aggregated robot task R_i loads a wafer at a PM for the process step, and the PM is unloaded by R_{i+1}, which also loads the wafer at a PM for the next process step. We now propose a necessary and sufficient condition for the tool to have no deadlock.

Theorem 1: Suppose that the Petri net model of a single-armed cluster tool with a reentrant wafer flow is marked by the initial marking rule. Then, the model is deadlock free if and only if for each reentrant PM

$$R_i \ll R_{i+1} \ll R_j \ll R_{j+1}$$

for any pair of process steps i and j such that they share the reentrant PM and $i < j$.

Proof: Since the initial marking rule prevents two possible deadlocks, types I or II, at any nonreentrant PM independently of the robot task sequence, it suffices to consider the sequence of robot tasks for reentrant process steps.

($\Leftarrow$) Suppose that the tool has a deadlock due to improper sequence of the robot tasks for reentrant process steps. We consider a type I deadlock that a PM is already occupied by a wafer when the robot tries to load a wafer into the PM. This occurs only when a robot task R_i that loaded a wafer into the PM is followed by another robot task R_j for loading a wafer for the process step j such that $i < j$ into the PM before the robot task R_{i+1} for unloading the wafer at the PM is performed, where the two process steps i and j share the PM. This implies that $R_j \ll R_{i+1}$ for some $i < j$. This contradicts condition (1). Therefore, there should be no type I deadlock.

We now examine a type II deadlock that a PM does not have the wafer that a specific robot task wishes to unload. We first consider the case that the PM has no wafer. This occurs only when a robot task R_{i+1} that unloaded a wafer from a PM is followed by another robot task R_{j+1} for unloading a wafer for the process step j such that $i < j$ from the PM before the robot task R_j for loading the wafer into the PM is performed. This implies that $R_{j+1} \ll R_j$ for some $i < j$. This contradicts condition (1). The other type II deadlock is that when a robot task R_{j+1} tries to unload a wafer that completes a specific next process step j from a PM, the process step i that has been completed for a wafer at the PM is not equal to j, that is, $i < j$ or $i > j$. Case $i < j$ implies that the next robot task of the robot task R_i for loading a wafer into the PM is not R_{i+1} but R_{j+1}. That is, $R_i \ll R_{j+1} \ll R_{i+1}$ or $R_{i+1} \ll R_i \ll R_{j+1}$ for some $i < j$. This violates condition (1). Likewise, the other case $i > j$ also implies that $R_i \ll R_{j+1} \ll R_{i+1}$ or $R_{i+1} \ll R_i \ll R_{j+1}$ for some $i > j$. This implies that $R_j \ll R_{i+1} \ll R_{j+1}$ or $R_{j+1} \ll R_j \ll R_{i+1}$ for some $i < j$. This also violates condition (1). Consequently, there should be no type II deadlock. Since any deadlock implies violation of condition (1), we conclude that if condition (1) holds, the tool model has no deadlock.

($\Rightarrow$) Robot tasks R_i, R_{i+1}, R_j, and R_{j+1} perform loading (i), unloading (i), loading (j), and unloading (j) tasks, respectively, where the number in a parenthesis indicates the corresponding process step. Suppose that for a reentrant PM, condition (1) is violated for some two process steps i and j such that they share the reentrant PM and $i < j$. In this case, $R_{i+1} \ll R_i$, $R_{j+1} \ll R_j$, or $R_j \ll R_{i+1}$. The first two cases trivially lead to a type II deadlock because they imply that a wafer tries to be unloaded before it is loaded. The last case also leads to a type I deadlock since robot task R_j loaded a wafer for the process step j such that $i < j$ to a PM before the wafer that is loaded for process step i is unloaded by robot task R_{i+1}. Therefore, any violation of condition (1) leads to a deadlock. Consequently, when there is no deadlock in the model, condition (1) should hold. ■

Remark 1: We can prove that our Petri net models are all asymmetric choice nets. For such an asymmetric choice net, there is a well-known necessary and sufficient condition for deadlock freedom based on traps and siphons and special structural properties of the net [17]. Therefore, we can verify deadlock freedom of the Petri net model for a given robot task sequence. However, since the deadlock-free condition should be verified for each possible robot task sequence, it is hard to develop a compact form of deadlock-free condition.

*Example 1 [**Wafer Flow Pattern** $(1^1, 1, 1^1)$]:* Fig. 7 illustrates the Petri net models for different sequences for wafer flow pattern $(1^1, 1, 1^1)$. We observe that the only deadlock-free robot task sequence is $\sigma^1 = (1, 2, 3, 4)$ for the wafer flow pattern. The sequence satisfies the deadlock-free condition (1). Any other sequence violates the condition and causes a deadlock. We note

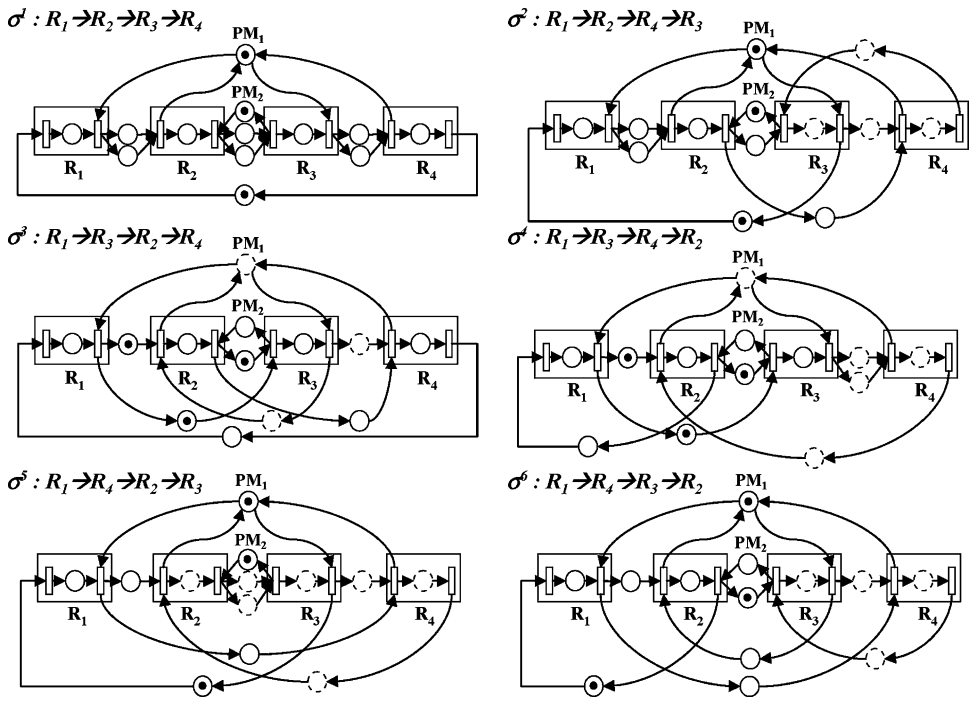


Fig. 7. Deadlock analysis of robot task sequences for wafer flow pattern $(1^1, 1, 1^1)$.

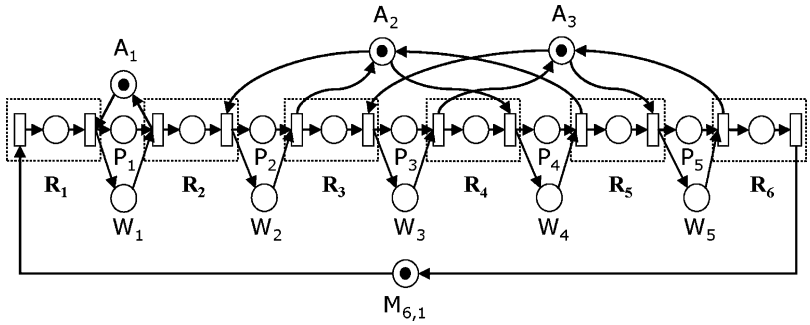


Fig. 8. Petri net model for deadlock-free sequence for wafer flow pattern $(1, 1^1, 1^2, 1^1, 1^2)$.

that the set of places marked with dotted circles in each net model is a siphon and also a trap, called a *siphon-and-trap*, of which presence is a necessary and sufficient condition for existence of a deadlock in an asymmetric choice net [17].

*Example 2 [**Wafer Flow Pattern** $(1, 1^1, 1^2, 1^1, 1^2)$]:* We present a more general, complicated flow pattern. ALD processes control the deposition thickness by repeating the deposition processes. Therefore, the sequence of deposition processes is repeated several times, even more than five. The first deposition layer requires three process steps: Al_2O_3 deposition process, Ta_2O_5 deposition process, and oxidation process. Each subsequent deposition layer repeats the last two process steps. Therefore, it is challenging to schedule a cluster tool with such complicated reentrant flow patterns. When two deposition layers are required, there are six aggregated robot tasks. Therefore, the number of possible robot task sequences amounts to 120 $(= 6!)$. From condition (1), we can identify the number of deadlock-free sequences to be ten. Fig. 8 illustrates the Petri net model for a deadlock-free sequence. As the number of deposition layers increases, the number of possible robot task sequences rapidly increases.

IV. Tool Cycle Time and Optimal Robot Task Sequence

We examine the tool cycle time for a given robot task sequence and the way of systematically determining the optimal deadlock-free robot task sequence and schedule. In a Petri net model for a robot task sequence for a reentrant wafer flow pattern, the availability place for a reentrant PM has as many input and output transitions as the number of process steps that share the PM. Such a place requires choice or decision on where output transition a token in the place should move. When each place has one and only one input and output transition, the Petri net is a decision-free event graph, a special class of asymmetric choice nets. For an event graph, the cycle time is efficiently computed as the critical circuit ratio. However, for an asymmetric choice net, the cycle time is not easily computed. For an asymmetric choice net, the cycle time or a bound on the cycle time is usually computed by complicated invariant analysis or reachability analysis [13]. However, such methods require nonpolynomial time algorithms of which computational complexity tends to be intractable, even for small sized nets. The size of our Petri net model grows fast as the number of process steps and the number of reentrances increase.

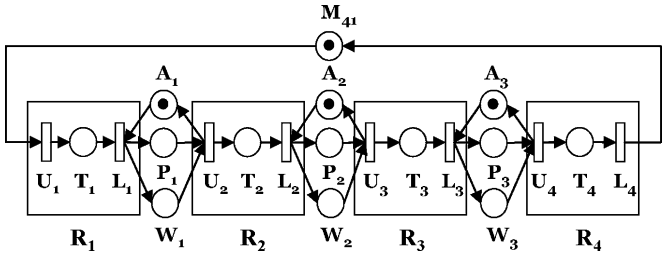


Fig. 9. Event graph model for $\sigma^1 = (1,2,3,4)$ of wafer flow pattern $(1^1,1,1^1)$.

We will show that although our Petri net models are asymmetric choice nets, due to a special structure, the Petri nets are behaviorally decision free; hence, the cycle time can be efficiently computed from the equivalent event graph. For a given Petri net $\mathcal{P}$ for a single-armed cluster tool with a deadlock-free sequence for a reentrant wafer flow, we let $\mathcal{E}(\mathcal{P})$ represent the event graph that is identical to the Petri net except that in the event graph each of the process steps that share a reentrant PM has its own separated PM availability place instead of sharing the single availability place for the reentrant PM. Fig. 9 illustrates such an event graph model for sequence $\sigma^1 = (1,2,3,4)$ of wafer flow pattern $(1^1,1,1^1)$. We observe that a pair of availability places A_1 and A_3 in the figure replaces the availability place A_1 for a reentrant PM in Fig. 2.

Theorem 2: A Petri net $\mathcal{P}$ for a deadlock-free robot task sequence of a reentrant wafer flow is behaviorally equivalent to the event graph $\mathcal{E}(\mathcal{P})$.

Proof: Consider a deadlock-free robot task sequence. From Theorem 1, for each of the process steps that share a reentrant PM, once the robot loaded a wafer into the PM, it should perform the unloading task before any other robot task for other process steps that share the PM. Therefore, the token at the availability place for the reentrant PM is used sequentially by the process steps that share the PM. After a process step returns the token, the next process step that shares the PM takes the token. At any time, the token that left at the availability place of the reentrant PM resides at most one of the processing places for the process steps that share the PM. There is no conflict between the process steps for using the token. Therefore, Petri net $\mathcal{P}$ is behaviorally decision free. Consequently, the availability place need not be shared by the processing places for the process steps. It suffices to make each processing place have its own availability place, which indicates the state that the process step is ready to start. Consequently, the firing sequence of the event graph $\mathcal{E}(\mathcal{P})$ is identical to the Petri net $\mathcal{P}$. ■

Theorem 2 suggests that the performance of Petri net $\mathcal{P}$ is identical to the performance of the event graph $\mathcal{E}(\mathcal{P})$. Therefore, the cycle time can be computed as the critical circuit ratio of the event graph $\mathcal{E}(\mathcal{P})$. The cycle time also can be computed from a linear program for the event graph [13], [15], [16]. Therefore, by extending the linear program appropriately, we can develop a mixed integer program for finding a deadlock-free robot task sequence that minimizes the cycle time. We introduce some notations. Let E indicate the set of arcs such that $(i,j) \in E$ if there is a place from transition T_i to transition T_j in the event graph $\mathcal{E}(\mathcal{P})$. When there are q places from T_i to T_j, we make copies of the arcs as many as q, that is, $(i^1,j^1),(i^2,j^2),\ldots,(i^q,j^q) \in E$. $A\ (\subset E)$ is the set of arcs that defines the wafer flow pattern and the PM work cycles. Consider the Petri net that excludes all places W_i and M_{ij} that are necessary for defining a robot task sequence. Then, if the net has a place from transition T_i to T_j, we let $(i,j) \in A$. $E - A$ is the set of arcs that defines a robot task sequence. The set is determined by the robot task sequence. d_i is the firing delay of transition T_i. h_{ij} and τ_{ij} are the holding time and the number of tokens of the place that corresponds to arc $(i,j) \in A$, respectively.

A firing schedule is defined by the set of firing epochs of the transitions $\{x_i^r | i \in T, r = 1,2,\ldots\}$, where T is a set of transitions and x_i^r is the rth firing epoch of transition T_i. When $x_i^{r+1} = x_i^r + \lambda$ for all $i \in T$ and $r = 1,2,\ldots$, the firing schedule is called *steady*. It is known that in an event graph, the minimum cycle time among the steady firing schedules is the same as the minimum cycle time among all firing schedules [15], [19]. Therefore, it suffices to consider only steady schedules. Since a steady schedule repeats an identical timing pattern for each firing cycle, we need to examine only a specific cycle. It is easily seen that in a steady schedule, for any place from transition T_i to T_j, $x_i^r = x_i^{r-k} + k\lambda$ for any $k = 0,1,2,\ldots$. We let $(x_1,x_2,\ldots,x_{|T|})$ be the schedule of a specific firing cycle r of a steady firing schedule. Consequently, the schedule should satisfy

$$x_j - x_i \geq d_j + h_{ij} - \tau_{ij}\lambda \quad \forall (i,j) \in A.$$

We note that the set of arcs or places for a robot task sequence $E - A$ is not determined yet. We let $u(i)$ and $l(i)$ indicate the indexes for transitions U_i and L_i, respectively. These should not be confused with u_i and l_i, which are the unloading and loading times for a PM for process step i. We let t_i and v be the transporting time for a PM for process step i and the move task time from a PM to other PM, respectively. The move times between each pair of PMs or the loadlocks are assumed to be identical. Let D be the set of disjunctive pairs of orderings between each pair of robot tasks R_is. A selection of order between a pair of disjunctive arcs (i,j) and (j,i) defines a place between R_i and R_j. We introduce an indicator function η_{ij} such that $\eta_{ij} = 1$ if $i+1 = j$; otherwise, $\eta_{ij} = 0$. Then, all possible robot task sequences can be represented by the disjunctive constraints

$$x_{u(j)} - x_{l(i)} \geq u_j + v\eta_{ij}$$
or
$$x_{u(i)} - x_{l(j)} \geq u_i + v\eta_{ji} \quad \forall ((i,j),(j,i)) \in D.$$

η_{ij} represents whether the robot moves or waits for performing R_j after completing R_i. The disjunctive constraints can be transformed into conjunctive ones using 0–1 variables δ_{ij}

$$x_{u(j)} - x_{l(i)} \geq u_j + v\eta_{ij} - (1-\delta_{ij})\lambda(l)$$
and
$$x_{u(i)} - x_{l(j)} \geq u_i + v\eta_{ji} - \delta_{ij}\lambda(l) \quad \forall ((i,j),(j,i)) \in D$$

TABLE I
DEADLOCK-FREE CONDITION

Precedence relationship	$\delta_{i,i+1}$	$\delta_{i+1,j}$	$\delta_{j,j+1}$	$\delta_{j+1,i}$	Sum
$R_i \ll R_{i+1} \ll R_j \ll R_{j+1}$	1	1	1	0	3
$R_{i+1} \ll R_j \ll R_{j+1} \ll R_i$	0	1	1	1	3
$R_j \ll R_{j+1} \ll R_i \ll R_{i+1}$	1	0	1	1	3
$R_{j+1} \ll R_i \ll R_{i+1} \ll R_j$	1	1	0	1	3

where $\delta_{ij} = 1$ if $x_{u(j)} - x_{l(i)} \geq u_j + v\eta_{ij}$ and $\delta_{ij} = 0$ if $x_{u(i)} - x_{l(j)} \geq u_i + v\eta_{ij}$.

Let m_i be the number of parallel PMs for process step i. The number of tokens in the processing circuit for process step i should be no more than m_i. Therefore, we have constraints on tokens

$$\tau_{l(i),u(i+1)} + \tau_{u(i+1),l(i)} \leq m_i \quad \forall i = 1, \ldots, n.$$

When a robot task sequence is chosen, an arc is selected from each pair of disjunctive arcs. Although the selected set of arcs S includes redundant arcs, they define the circuit for the corresponding robot work cycle. The arc set $E - A$ that defines a robot task sequence in the Petri net model is then considered as the one that is made from S by removing all redundant arcs. We note that each of the robot waiting places W_i and the move places M_{ij} in a Petri net model corresponds to each arc in the set $E - A$. Once $E - A$ is determined, each place that corresponds to an arc of $E - A$ is generated in the resulting Petri net model. It can be seen that the places corresponding to the redundant arcs do not change the behavior of the Petri net model. Since there is only one robot arm, there is only one token in the robot work circuit, which indicates the availability of the robot arm. The initial marking rule assigns the token at the input place of R_1 in the robot work circuit. All other places for the arcs of $E - A$ that define the robot work cycle should not have any token. Also, the PM for the first process step should be available, that is, the corresponding place should not have any token. Therefore, we have the following:

$$\begin{aligned} &\tau_{u(i),l(1)} = 1 \quad \forall i \neq 1 \\ &\tau_{u(i),l(j)} = 0 \quad \forall j \neq 1, i \\ &\tau_{l(1),u(2)} = 0. \end{aligned}$$

Let R be the set of indexes for the aggregated robot tasks for loading a wafer at reetranct PMs. We wish to add the deadlock-free condition (1) into the constraints. To do this, we have the following result.

Corollary 1: The deadlock-free condition (1) holds if and only if

$$\delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} = 3 \quad \forall (i,j) \in R. \qquad (1)$$

Proof: Once the robot task sequence is determined, the robot repeats a cyclic sequence of robot tasks. Therefore, the precedence relationship $R_i \ll R_{i+1} \ll R_j \ll R_{j+1}$ of condition (1) also holds cyclically. In other words, the four precedence relationships in Table I are all equivalent. Depending on how a cycle is taken, one of them holds. In each case, the instances of the four robot tasks are considered to belong to the same cycle. Each case of the precedence relationship in the table is equivalently represented by appropriate value assignment to the corresponding sequencing decision variables $\delta_{i,i+1}$, $\delta_{i+1,j}$, $\delta_{j,j+1}$, and $\delta_{j+1,i}$. Table I shows such value assignment. It is seen that when for each case of the precedence relations, the sum is 3. Therefore, if condition (1) holds, $\delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} = 3$.

Now, we suppose that $\delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} \neq 3$. When the sum is 4, $\delta_{i,i+1} = \delta_{i+1,j} = \delta_{j,j+1} = \delta_{j+1,i} = 1$. This implies a circular order between the instances of the robot tasks in the same cycle. This is an infeasible sequence and hence condition (1) does not hold. When $\delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} < 3$, more than one of the sequencing decision variables are zero. This indicates that at least a pair of robot tasks in each case of the four precedence relationships should be reversed. This violates condition (1). Consequently, condition (1) is equivalent to condition (1). ■

By collecting the constraints, we now have an optimization model as follows:

$$\begin{aligned} &\min \quad \lambda \\ &\text{subject to} \\ &\quad x_j - x_i \geq d_j + h_{ij} - \tau_{ij}\lambda \qquad \forall (i,j) \in A \\ &\quad x_{u(j)} - x_{l(i)} \geq u_j + v\eta_{ij} - (1 - \delta_{ij})\lambda \\ &\text{and} \\ &\quad x_{u(i)} - x_{l(j)} \geq u_i + v\eta_{ji} - \delta_{ij}\lambda \qquad \forall ((i,j),(j,i)) \in D \\ &\quad \delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} = 3 \qquad \forall (i,j) \in R \\ &\quad \tau_{l(i),u(i+1)} + \tau_{u(i+1),l(i)} \leq m_i \qquad \forall i = 1, \ldots, n \\ &\quad \tau_{u(i),l(1)} = 1 \qquad \forall i \neq 1 \\ &\quad \tau_{u(i),l(j)} = 0 \qquad \forall j \neq 1, i \\ &\quad \tau_{l(1),u(2)} = 0, \\ &\quad \lambda \geq 0,\ x_i \geq 0,\ \delta_{ij} = 0 \text{ or } 1,\ \lambda \in \mathbb{R},\ x_i \in \mathbb{R}, \end{aligned}$$

where λ, x_i, τ_{ij}, and δ_{ij} are decision variables. Symbol $\mathbb{R}$ indicates the set of real numbers.

The first and second constraints have nonlinear terms $\tau_{ij}\lambda$ and $\delta_{ij}\lambda$. They can be linearized by dividing all terms by λ and redefining the decision variables as the divided terms. By letting $\mu \equiv 1/\lambda$ and $y_i \equiv x_i/\lambda$, the above model can be linearized into a mixed integer program (MIP) as follows:

$$\begin{aligned} &MIP: \qquad \max \quad \mu \\ &\text{subject to} \\ &\quad y_j - y_i \geq (d_j + h_{ij})\mu - \tau_{ij} \quad \forall (i,j) \in A \\ &\quad y_{u(j)} - y_{l(i)} \geq (u_j + v\eta_{ij})\mu - (1 - \delta_{ij}) \end{aligned}$$

and

$$
\begin{aligned}
&y_{u(i)} - y_{l(j)} \geq (u_i + v\eta_{ji})\mu - \delta_{ij} \qquad \forall ((i,j),(j,i)) \in D \\
&\delta_{i,i+1} + \delta_{i+1,j} + \delta_{j,j+1} + \delta_{j+1,i} = 3 \qquad \forall (i,j) \in R \\
&\tau_{l(i),u(i+1)} + \tau_{u(i+1),l(i)} \leq m_i \qquad \forall i = 1, \ldots, n \\
&\tau_{u(i),l(1)} = 1 \qquad \forall i \neq 1 \\
&\tau_{u(i),l(j)} = 0 \qquad \forall j \neq 1, i \\
&\tau_{l(1),u(2)} = 0, \\
&\mu \geq 0,\ y_i \geq 0,\ \delta_{ij} = 0 \text{ or } 1,\ \mu \in \mathbb{R},\ y_i \in \mathbb{R}
\end{aligned}
$$

where μ, y_i, τ_{ij}, and δ_{ij} are decision variables.

The MIP model finds an optimal deadlock-free robot task sequence for a given wafer flow pattern. It also determines a steady optimal schedule, that is, the timings of the tasks. Additional linear constraints can be incorporated. For instance, a time window constraint on the allowable wafer delays within the processing chambers can be added in order to prevent quality problems due to residual gases and heat. Such a time window constraint follows:

$$x_{l(i)} - x_{u(i+1)} \geq -(p_i + u_{i+1} + b_i) + \tau_{l(i),u(i+1)}\lambda \ \forall i = 1, \ldots, n \tag{2}$$

where p_i is the process time for process step i and b_i is the maximum allowable wafer delay within the processing chamber after completing process step i.

V. Application to Cluster Tool Engineering

We now present how our results are used for cluster tool engineering, which determines the number of PMs for each process step, the optimal robot task sequence, and the schedule with the minimum cycle time. To do this, we introduce the notion of workload for each process step since the workload measure often plays an important role in flow-type manufacturing systems such as assembly lines or flexible flow lines. A conventional workload measure for a process step is the sum of the process times of the operations performed for the process step. This determines the bottleneck and the cycle time in conventional assembly lines and gives a lower bound for the cycle time in most flow-type manufacturing lines. In a cluster tool, the wafers go through series-parallel process steps, but the wafer transfer between the process steps is limited by the robot. Lee *et al.* [7] extended the workload measure for process step i in a single-armed cluster tool as

$$w_i \equiv \frac{p_i + 2(u_i + t_i + l_i) + v}{m_i}. \tag{3}$$

It is because two aggregated robot tasks, R_i and R_{i+1}, and one empty move as well as the processing operation at a PM are required for a wafer to take a process step i in a backward sequence $(1, n+1, n, n-1, \ldots, 2)$. For any other sequence, the two robot tasks are required for completing the processing of a wafer at a PM. It is known that balancing the workloads of the process steps gives many desirable properties for cluster tool operation and scheduling [7], [8]. Minimizing the maximum workload of the process steps also minimizes the tool cycle time.

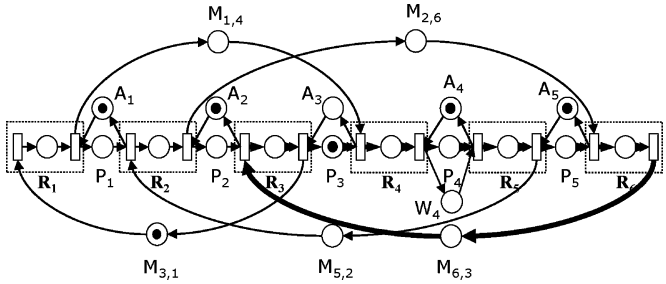


Fig. 10. Event graph model for $\sigma = (1, 4, 5, 2, 6, 3)$.

The popular backward sequence for a single-armed cluster tool with no reentrance is optimal when the workloads between the process steps are reasonably balanced. The wafer delays within a PM's chamber after processing at the PM are also minimized by balancing the workload. We therefore extend the workload measure for cluster tools with reentrant wafer flows. Let $\Lambda(i)$ denote the set of all process steps that share a reentrant PM that performs process step i. For a nonreentrant process step i, $\Lambda(i) = \{i\}$. For a cluster tool with reentrant wafer flows, the workload for a PM that performs process step i is defined to be

$$w_i \equiv \frac{\sum_{j \in \Lambda(i)} (p_j + 2(u_j + t_j + l_j) + v)}{m_j}. \tag{4}$$

The sum of the process times for the process steps that share a set of PMs is divided by the number of the shared PMs. Therefore, this workload definition is general and applicable even when multiple PMs are assigned to multiple identical processes regardless of the PM operating strategy, which was explained in Section II. For the case that has no reentrant wafer flow, this definition is equivalent to the original workload definition in (3).

For a cluster tool with reentrant wafer flows, the workloads can be balanced by allocating the process steps to each PM, that is, by controlling the number of reentrances for each reentrant PM, accommodating the number of parallel PMs, or modifying the process times within the technically allowable range. These are all important engineering decisions.

Each robot work cycle performs aggregated robot tasks and empty moves at least $n+1$ times. Thus, the workload of the robot itself is $\sum_{i=1}^{n+1}(u_i + t_i + l_i + v)$. The cycle time of a PM for each process step i tends to be mostly determined by its workload $w_i = (\sum_{j \in \Lambda(i)}(p_j + 2(u_j + t_j + l_j) + v))/m_j$; although, the exact cycle time of a PM depends on the circuits in the Petri net model that are completely determined by the robot task sequence. By considering the robot as another resource, the combined maximum workload measure is

$$W \equiv \max \left\{ \max_{i=1,\ldots,n} \frac{\sum_{j \in \Lambda(i)} (p_j + 2(u + t + l) + v)}{m_j}, \sum_{i=1}^{n+1} (u_i + t_i + l_i + v) \right\}. \tag{5}$$

The maximum workload measure W can be computed regardless of the robot task sequencing decision, which requires running a combinatorial algorithm using the tool data such as the

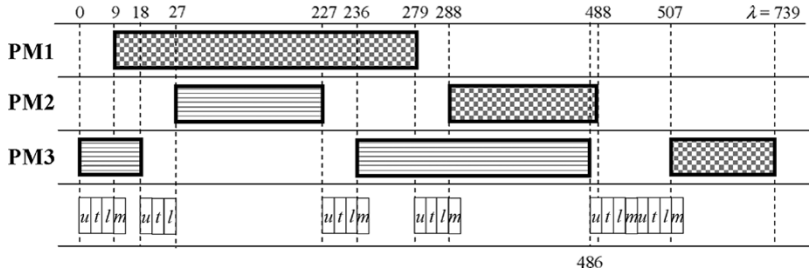


Fig. 11. Optimal schedule.

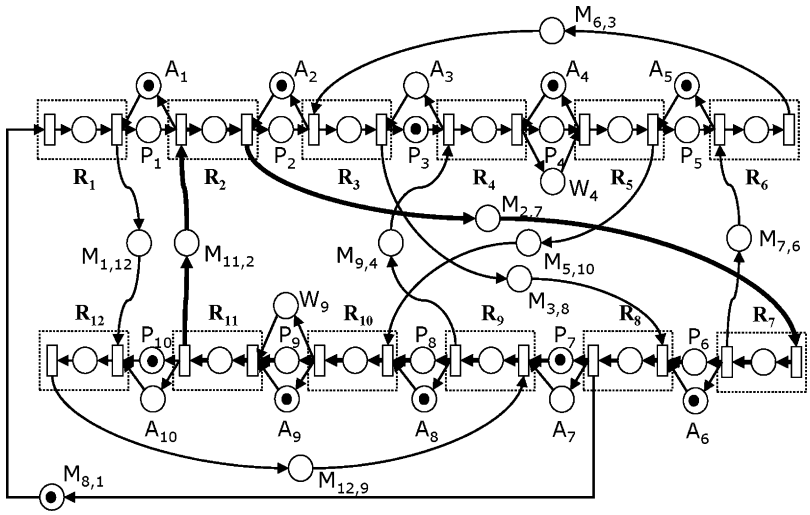


Fig. 12. Event graph model for sharing strategy.

process times, the robot task times, the number of parallel PMs, and so on. The maximum workload measure evaluates the robot workload as well as the workloads of the PMs. When the robot workload is not the bottleneck, as in most practical cases, W measures the balance of the workloads of the PMs. W is also a lower bound of the tool cycle time since the tool cycle time becomes larger than the workload due to interference among the work cycles of the PMs and the robot. Such a sequence-independent workload measure is useful for roughly evaluating the tool design alternatives on the number of PMs for each process step and the process times. Without deliberately determining the optimal robot task sequence, we can quickly compare the approximate cycle times of the alternatives and obtain an insight on how to improve the decisions.

Example 3: We consider the single-armed cluster tool model in Example 2 in Section III. The tool has three PMs for five process steps. The wafer flow pattern $(1, 1^1, 1^2, 1^1, 1^2)$ indicates Loadlock $\rightarrow$ PM1 $\rightarrow$ PM2 $\rightarrow$ PM3 $\rightarrow$ PM2$\rightarrow$ PM3 $\rightarrow$ Loadlock. The processing times of PM1, PM2, and PM3 are 270, 200, and 250 s, respectively. Process steps 2 and 3 have the same processing times as process steps 4 and 5, respectively. The times for unloading, loading, transporting, and move tasks (u_i, l_i, t_i, and m_i) are all 3 s. By solving the MIP model, we have the optimal robot task sequence as $\sigma^* = (1, 4, 5, 2, 6, 3)$, that is, $R_1 \rightarrow R_4 \rightarrow R_5 \rightarrow R_2 \rightarrow R_6 \rightarrow R_3 \rightarrow R_1$ and the optimal cycle time 739 s. The resulting event-graph model and the schedule are shown in Figs. 10 and 11, respectively. The circuit that is represented by bold arcs is the *critical circuit* with the maximum circuit ratio, which is the same as the minimum cycle time. Fig. 11 shows the schedule for a single cycle, one half for the schedule for a wafer and the other part for another wafer. The boxes with the identical shading indicate the process times of the same wafer. The schedule is repeated for each cycle.

The maximum workload W is 542 s. Interference between the work cycles increases the cycle time as much as 197 s $(= 739\text{ s} - 542\text{ s})$. The robot work cycle limits the PM work cycles. The workloads of PM1, PM2, and PM3 are 291, 442, and 542 s, respectively. The robot workload is just 72 s. PM3 for process steps 3 and 5 is the bottleneck and the workloads are significantly imbalanced. We therefore add an additional PM, PM4, for the two identical process steps. Due to the additional PM, the workload for PM3 reduces to 271 s. The maximum workload W is now 442 s. There are two alternative strategies for operating the two PMs, PM3 and PM4, *sharing* and *dedicating*, as proposed in Section II. First, the sharing strategy makes PM3 and PM4 share process steps 3 and 5. A wafer visits the PM that it visited first. That is, the sharing strategy keeps the reentrance in view of wafers and increases the number of parallel PMs for reentrant process steps 3 and 5. Therefore, the wafer flow pattern becomes $(1, 1^1, 2^2, 1^1, 2^2)$. Second, the dedicating strategy makes PM3 and PM4 take charge of process steps 3 and 5, respectively. The wafer flow pattern is $(1, 1^1, 1, 1^1, 1)$. In this case, no PM has reentrance. The dedicating strategy is thus considered to relax the complication or constraints due to the reentrance wafer flow and sharing a PM by multiple process steps. Since reentrance tends to make the performance significantly worse, we expect that the dedicating strategy will give a much shorter cycle time. Fig. 12 shows the event graph model

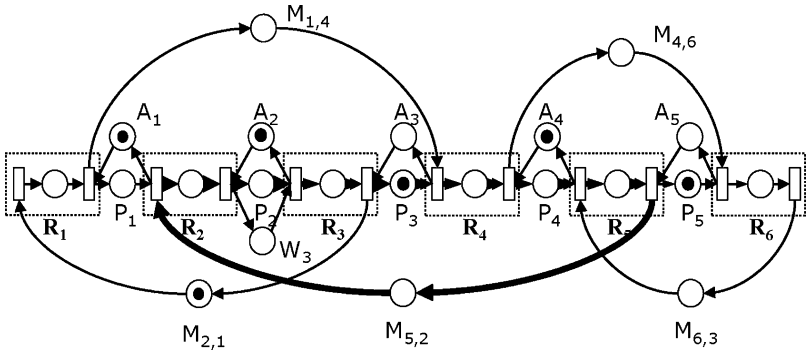


Fig. 13. Event graph model for dedicating strategy.

TABLE II
COMPARISON OF SHARING AND DEDICATING STRATEGIES

Process times	Wafer flow pattern	Strategy	Cycle time	Max. W/L
(270,200,250,200,250)	$(1, 1^1, 1^2, 1^1, 1^2)$		739	542
	$(1, 1^1, 2^2, 1^1, 2^2)$	(S)	490	442
	$(1, 1^1, 1, 1^1, 1)$	(D)	689	442
	$(1, 2^1, 1^2, 2^1, 1^2)$	(S)	542	542
	$(1, 1, 1^1, 1, 1^1)$	(D)	739	542
(200,200,200,200,200)	$(1, 1^1, 1^2, 1^1, 1^2)$		639	442
	$(1, 1^1, 2^2, 1^1, 2^2)$	(S)	442	442
	$(1, 1^1, 1, 1^1, 1)$	(D)	639	442
	$(1, 2^1, 1^2, 2^1, 1^2)$	(S)	442	442
	$(1, 1, 1^1, 1, 1^1)$	(D)	639	442
(500,200,200,200,200)	$(1, 1^1, 1^2, 1^1, 1^2)$		639	521
	$(1, 1^1, 2^2, 1^1, 2^2)$	(S)	521	521
	$(1, 1^1, 1, 1^1, 1)$	(D)	639	521
	$(1, 2^1, 1^2, 2^1, 1^2)$	(S)	521	521
	$(1, 1, 1^1, 1, 1^1)$	(D)	639	521

for the sharing strategy with optimal robot task sequence $\sigma^* = (1, 12, 9, 4, 5, 10, 11, 2, 7, 6, 3, 8)$. For the sharing strategy, the cycle time is 490 s, which is now close to the maximum workload 442 s. For the dedicating strategy, the optimal robot task sequence is $\sigma^* = (1, 4, 6, 5, 2, 3)$. The event graph is shown in Fig. 13. For the case, the cycle time is 689 s. This is smaller than the original cycle time but much larger than the cycle time for the sharing model. This does not coincide with our common sense that the dedicating strategy, relaxing complication of reentrance for process steps 3 and 5, has a much longer cycle time than the sharing strategy that keeps the reentrance constraint.

In order to further understand the bizarre phenomenon for the two strategies, we examine more cases for the tool model in Table II. In Table II, *process time* indicates the process time of each process step. For instance, process time case (270, 200, 250, 200, 250) means that the process times of process steps 1, 2, 3, 4, and 5 are 270, 200, 250, 200, and 250 s, respectively. The wafer flow patterns in the second and third rows for each process time case are for the case of adding PM4 to process steps 3 and 5, respectively. The wafer flow patterns for the fourth and fifth rows indicate the case for adding PM4 to process steps 2 and 4 that have shared PM2, instead of process steps 3 and 5. (S) and (D) after each wafer flow pattern indicate the sharing and dedicating strategies, respectively. We observe that for different process time cases, the sharing strategy is always better than the dedicating strategy. A possible reason is that although the sharing strategy keeps the reentrance constraint, it requires that the two PMs be shared by the two process steps. It seems that this resource sharing, as usual resource pooling methods,

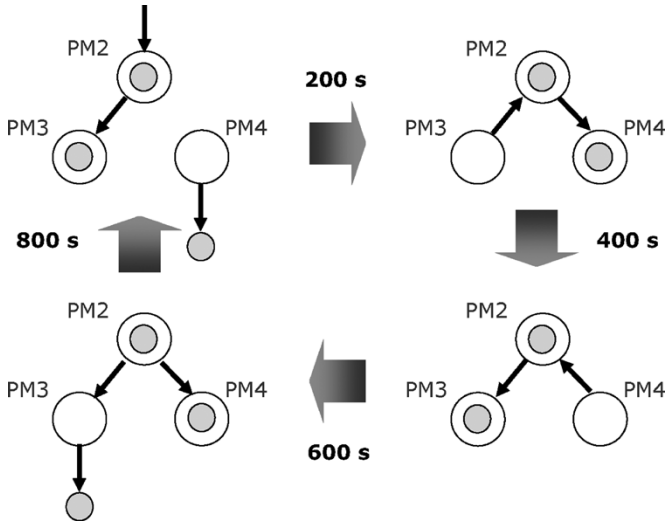


Fig. 14. State change by sharing strategy.

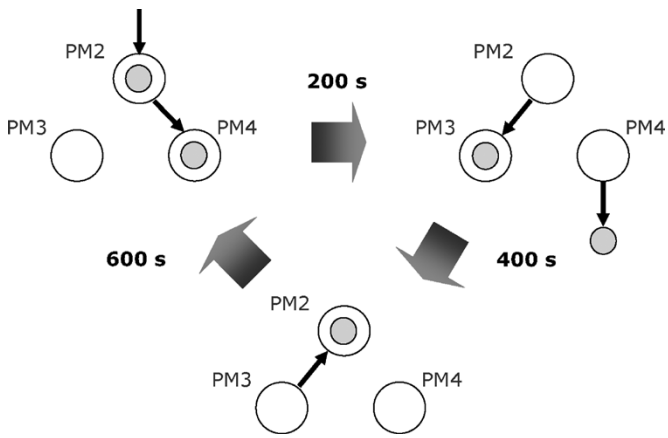


Fig. 15. State change by dedicating strategy.

improves the performance. The state changes for the sharing and dedicating strategies are illustrated in Figs. 14 and 15, respectively. Without loss of generality, we consider a simpler model with omitting PM1 for simplicity. In the model, PM2 performs the first process step. Then, the wafer flow patterns for the sharing and dedicating strategies are $(1^1, 2^2, 1^1, 2^2)$ and

$(1^1, 1, 1^1, 1)$, respectively. We assume that the process times of all PMs are all 200 s. An arrow in figures indicates a trace of the wafer movement. The sharing strategy keeps PM2 busy, as shown in Fig. 14. The dedicating strategy causes idle time at PM2, as shown in Fig. 15. That is, the sharing strategy better utilizes the additional PM, PM2. The sharing strategy has cycle time 400 s, much shorter than the cycle time 600 s of the dedicating strategy. Consequently, when we add a PM for a reentrant PM, the sharing strategy is much more efficient. However, this may not always be true. Suppose that we add two PMs, one for each reentrant PM. The dedicating strategy then eliminates the reentrances completely and makes the wafer flow pattern be (1, 1, 1, 1, 1). Then, the well-known backward sequence can be used [7] and the dedicating strategy was found to be better. When there is only one reentrant PM, the dedicating strategy is found to be better.

VI. Conclusion

We proposed a systematic Petri net modeling method for single-armed cluster tools with various reentrant wafer flow patterns. The key modeling strategy is to model the process steps separately and let the work cycles of the reentrant process steps share the availability token for a reentrant PM. We developed a simple necessary and sufficient condition for preventing a deadlock. We showed that the Petri net models, although they are asymmetric choice nets, reduce to equivalent event graphs for which the cycle time is easily computed as the critical circuit ratio. From the results, we developed an MIP model for determining the optimal robot task sequence, the minimum cycle time, and the optimal schedule. We introduced two different strategies, sharing and dedicating, for using multiple PMs for identical process steps. We also extended a workload measure for cluster tools with reentrant wafer flows. Finally, we discussed how the results are used for engineering a cluster tool. We found that depending on the PM operating strategy, the cycle time can be significantly different.

Our proposed method of modeling a cluster tool as an event graph and deriving a mixed integer programming model from the event graph can be used for other cluster tools. The key is to derive the decision-free event graph model by examining the tool behavior and to identify how the sequencing decision affects the arcs or precedence relations between the events or transitions of the event graph model. We have seen that such a methodology is effective even for complicated cluster tool models due to reentrant wafer flows.

Although the MIP model determines the steady schedule as well as the optimal robot task sequence, the earliest starting schedules, steady or not, that start every operation as soon as the preceding ones complete but have the same cycle time as the steady schedule can be computed through the steady-state analysis based on the minimax algebra [15]. Such a schedule can be easily implemented by event-based control rather than deliberate timing control. Further topics for reentrant cluster tools include examining properties of the wafer residency times and scheduling under the wafer residency time constraints and further implication of workload balancing on sequencing and wafer delays. We need to examine further the deadlocks and robot task sequencing of dual-armed cluster tools with reentrant wafer flows for which conventional swap operation cannot be used.

References

[1] S. C. Wood, S. Tripathi, and F. Moghadam, "A generic model for cluster tool throughput time and capacity," in *Proc. IEEE/SEMI Advanced Semiconductor Manufacturing Conf.*, 1994, pp. 194–199.

[2] T. L. Perkinson, P. K. McLarty, and R. S. Gyurcsk, "Single-wafer cluster tool performance: an analysis of throughput," *IEEE Trans. Semiconduct. Manufact.*, vol. 7, no. 3, pp. 369–373, Aug. 1994.

[3] T. L. Perkinson and R. S. Gyurcsik, "Single-wafer cluster tool performance: An analysis of the effects of redundant chambers and revisitation sequences on throughput," *IEEE Trans. Semiconduct. Manufact.*, vol. 9, no. 3, pp. 384–400, Aug. 1996.

[4] S. Venkatesh, R. Davenport, P. Foxhoven, and J. Nulman, "A steady-state throughput analysis of cluster tools: dual-blade versus single-blade robots," *IEEE Trans. Semiconduct. Manufact.*, vol. 10, no. 4, pp. 418–424, Nov. 1997.

[5] W. M. Zuberek, "Cluster tools with chamber revisiting—Modeling and analysis using timed Petri nets," *IEEE Trans. Semiconduct. Manufact.*, vol. 17, no. 3, pp. 333–344, Aug. 2004.

[6] Y.-H. Shin, T.-E. Lee, J.-H. Kim, and H.-Y. Lee, "Modeling and implementing of a real-time scheduler for dual-armed cluster tools," *Comput. Industry*, vol. 45, no. 1, pp. 13–27, 2001.

[7] T.-E. Lee, H.-Y. Lee, and Y.-H. Shin, "Workload balancing and scheduling of a single-amred cluster tool," in *Proc. 5th APIEMS Conf.*, Gold Coast, Australia, 2004, pp. 1–15.

[8] J.-H. Kim, T.-E. Lee, H.-Y. Lee, and D.-B. Park, "Scheduling analysis of time-constrained dual-armed cluster tools," *IEEE Trans. Semiconduct. Manufact.*, vol. 16, no. 3, pp. 521–534, Aug. 2003.

[9] T.-E. Lee and S.-H. Park, "An extended event graph with negative places and tokens for time window constraints," *IEEE Trans. Automat. Sci. Eng.*, vol. 2, no. 4, pp. 319–332, 2005.

[10] J. Liu, Y. Jiang, and Z. Zhou, "Cyclic scheduling of a single hoist in extended electroplating lines: A comprehensive integer programming solution," *IIE Trans.*, vol. 34, pp. 905–914, 2002.

[11] M. A. Manier, C. Varnier, and P. Baptiste, "Constraint-based model for the cyclic multi-hoist scheduling problem," *Production Planning Contr.*, vol. 11, no. 3, pp. 244–257, 2000.

[12] S.-G. Kim, "Operating strategies for dual-armed cluster tools with reentrant job flows," M.S. thesis, Dept. Industrial Engineering, Korea Advanced Institute of Science and Technology, Taejon, Korea, 2004.

[13] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[14] T.-E. Lee and M. E. Posner, "Performance measures and schedules in periodic job shops," *Oper. Res.*, vol. 45, no. 1, pp. 72–91, 1997.

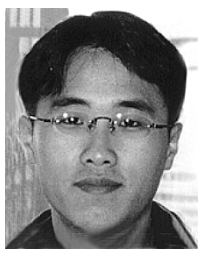
[15] T.-E. Lee, "Stable earliest schedule for cyclic job shops: A linear system approach," *Int. J. Flexible Manufact. Syst.*, vol. 1, pp. 59–82, 2000.

[16] J.-W. Seo and T.-E. Lee, "Steady state analysis of cyclic job shops with overtaking," *Int. J. Flexible Manufact. Syst.*, vol. 14, no. 4, pp. 291–318, 2002.

[17] K. Barkaoui and J.-F. Pradat-Peyre, "On liveness and controlled siphons in Petri nets," *Lecture Notes Computer Sci.*, vol. 1091, pp. 57–72, 1996.

[18] M. Jeng, "Comments on "timed Petri nets in modeling and analysis of cluster tools"," *IEEE Trans. Automat. Sci. Eng.*, vol. 2, no. 1, pp. 92–93, 2005.

[19] F. L. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*. New York: Wiley, 1992.



Hwan-Yong Lee received the B.S., M.S., and Ph.D. degrees, all in industrial engineering, from the Korea Advanced Institute of Science and Technology (KAIST).

Since 2005, he has been with LCD Business Division, Samsung Electronics. His research interests include scheduling and control of automated semiconductor manufacturing systems and quality management of LCD manufacturing.

Tae-Eog Lee received the Ph.D. degree in industrial and systems engineering from The Ohio State University, Columbus, in 1991.

He is a Professor in the Department of Industrial Engineering, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea. His research interests include modeling, scheduling, control, and control software architecture for automated manufacturing systems such as cluster tools and track equipment for semiconductor manufacturing and discrete-event systems.

Dr. Lee is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.