INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 17, NO. 4, DECEMBER 2012,113-123

Scheduling Single-Arm Multi-Cluster Tools with Lower Bound Cycle Time via Petri Nets

QingHua ZHU and Yan QIAO

Abstract- **Multi-cluster tools are widely adopted in wafer fabrication. It is of paramount importance to schedule them to achieve their optimal throughput. This work intends to find their one-wafer optimal periodic schedule to attain their lower bound cycle time. A resource-oriented Petri net model is developed to describe not only their steady state behavior but also behavior of their initial and final transient processes. Based on the model, this work derives the conditions under which one-wafer cyclic schedule with lower bound of cycle exists. Then, an algorithm is given to find it by scheduling the individual cluster tools one by one, which requires simple calculation only. Also, based on the Petri net model, an effective method for the implementation of such an optimal schedule is proposed. Examples are given to show the results.**

Index Terms—**semiconductor manufacturing, cluster tool, scheduling, Petri net**

1. INTRODUCTION

With the single-wafer processing technology, cluster tools are widely adopted in semiconductor manufacturing. A single cluster tool is composed of 4-6 process modules (PM), one wafer-delivering robot (R), and two loadlocks (LL) for cassette loading/unloading. The robot can be a single or dual-arm one, and thus, the corresponding tool is called single or dual-arm cluster tool. To improve the productivity, many manufacturers now integrate several cluster tools into a wafer fabrication system called a multi-cluster tool as illustrated in Fig. 1. A multi-cluster tool formed by $K \geq 2$ single-cluster tools is called K-cluster tools. A buffer with a highly limited wafer capacity for holding incoming and outgoing wafers is used between two adjacent single-cluster tools.

Extensive studies have been done in the modeling and performance evaluation of single-cluster tools [11, 13-18, 20-22; 27-31, 36]. It is found that, for a single wafer product processing in high-volume, a cluster tool operates under a steady state and thus a periodic schedule is the most desired. Under its steady state, a cluster tool operates in one of the two regions: transport- and process-bound ones. In the former, its robot is always busy and the system cycle time is determined by the robot task time. In the latter, its robot has idle time and the cycle time is decided by the processing time in PMs. Its robot moving time from one PM to another is often treated as a constant and is much shorter than the wafer processing time [4, 9]. In this case, a backward scheduling strategy is optimal for single-arm cluster tools [3, 10, 12].

Many wafer fabrication processes require that a wafer should be removed from a PM within a given time interval after it is completed [9, 11]. [20, 28] studied such a residency time constraint problem. The developed Petri net models are independent of the wafer flow pattern and thus reusable. By these models, the robot waiting is modeled such that to find an optimal schedule is to simply determine the robot waiting time. Determining it is also a focus in scheduling a multi-cluster tool, to be shown later.

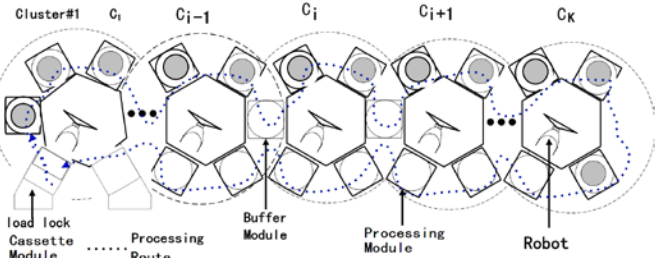


Fig. 1. The illustration of a multi-cluster tool

The use of multi-cluster tools in industry can be traced back to a decade ago [7, 8]. However, a few of research results have been reported. In [5], an event graph model, as a special class of Petri nets, is used to describe the dynamic behavior of a multi-cluster tool. Based on it, a search method is proposed to find an optimal periodic schedule, which is not necessarily a one-wafer cyclic one.

The minimal one-wafer cycle time of an individual cluster tool is called a fundamental period (FP) [34]. To reduce the computational complexity, without considering the robot moving time, a decomposition method is proposed in [33, 34]. A K-cluster tool is decomposed into K single-cluster tools and the FP for each single-cluster tool can be obtained as done for scheduling single-cluster tools. Then, by analyzing time delays needed for loading and unloading the shared buffers between the cluster tools in a multi-cluster tool, the cycle time for the system is determined. In this way, a feasible schedule is found.

With robot moving time considered, a polynomial algorithm is presented to find an "optimal" multi-wafer schedule for multi-cluster tools in [1, 2]. A key idea is to determine the number of wafers that are concurrently processed in each individual cluster tool in the system via a polynomial algorithm. Next, the schedule of an entire multi-cluster tool can be obtained by another polynomial algorithm, regardless any individual tool is process or

This work was supported in part by the National Natural Science Foundation of China under Grant 60974098.

Q. H. Zhu is with the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China (e-mail: zhuqh@gdut.edu.cn).

Y. Qiao is with the Department of Industrial Engineering, School of Electro-Mechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China (e-mail: dr.yqiao@gmail.com).

transport-bound, representing a significant advancement in scheduling multi-cluster tools.

Let FP_i be the fundamental period of the i-th tool in a K-cluster tool. Then, $\Pi = \max\{FP_1, FP_2, \ldots, FP_K\}$ is its lower bound of cycle time. Clearly, it is the most desired if it is scheduled such that its cycle time is Π.

The method presented in [1, 2] enable us to find an optimal cyclic schedule but it is in general a multi-wafer schedule. Their work did not answer under what condition an optimal one-wafer cyclic schedule can be found with cycle time Π. Note that a schedule of a K-cluster tool may be optimal, i.e., its cycle time cannot be lower; but cannot be as low as Π.

This gives rise to a question whether there is always a backward schedule for a K-cluster tool such that the lower bound of cycle time is reached? If not, under what condition and how such a schedule can be found? This work answers them for the first time.

It is known that a one-wafer cyclic schedule is easier to understand, control, and implement to keep uniform wafer quality [3, 6]. Hence, it is desired that a one-wafer cyclic schedule is found for a multi-cluster tool. Further, if such a schedule exists, it is the most desired that the lower bound of cycle time is reached. Yi *et al*. [34] give the conditions and schedules to realize Π, whereas they ignore the robots' moving time, which is necessary to predict the throughput from the practical view. However, when considering robots' moving time, Chan *et al.* [1] believe that, due to the tool interactions, some sort of K-cluster tools are unlikely to reach their Π by a one-wafer cyclic schedule. Thus, this work attempts to find it if existing. Notice that, in cluster tools, the robot moving time from one PM to another is much shorter than the wafer processing time [9], a cluster tool often operates in a process-bound region. Hence, this paper focuses on the cases that the bottleneck single-cluster tool in a multi-cluster is process-bound. Such a K-cluster tool is called process- dominant. The key to schedule a multi-cluster tool is how to coordinate the activities of its multiple robots. The problem is modeled by a generic Petri net (PN) model. With this model, the conditions under which one-wafer cyclic schedule reaching cycle time Π exists are derived. Then an efficient algorithm is given to find it if existing.

The remainder of the paper is organized as follows. After briefly discussing the multi-cluster tool operation, a PN model is developed in Section 2. Section 3 presents how the individual cluster tools should be scheduled. Then, the optimality conditions and scheduling algorithm for a multi-cluster tool are derived in Section 4. Illustrative examples are given in Section 5. Section 6 presents the conclusions

2. MODELING OF MULTI-CLUSTER TOOLS

2.1 Multi-cluster Tools

A K-cluster tool considered in this paper is composed of single-cluster tools that have single-arm robots. According to [1, 2], topologically, a multi-cluster tool contains no cluster tool cycle, and is tree-like. For the sake of simplicity in presentation, we address multi-cluster tools with a linear topology as shown in Fig. 1. Without loss of generality, following [1, 2], we assume that:

1) With two loadlocks, while one lot of wafers in one loadlock is being processed, the other loadlock can be used for loading/unloading another lot of wafers. In this way, a multi-cluster tool can be operated consecutively without being interrupted. Thus, it operates in a steady state for the identical wafer processing, or there are always wafers for processing;
2) Tools can hold one wafer at a time without a processing function;
3) A PM can process one wafer at a time; and
4) All the wafers in a cassette are processed in an identical sequence specified in the recipe and visit a PM no more than once (except for a buffer module); and the loading, unloading, moving time of robots and processing time of a wafer at a PM are deterministic.

Let $\mathbf{N}_n = \{1, 2, ..., n\}$ and $\Omega_n = \{0\} \cup \mathbf{N}_n$. As shown in Fig. 1, the K cluster tools forming a K-cluster tool are denoted by C_1, C_2, …, and C_K ($K \geq 2$). A buffer module (BM) shared by C_{i-1} and C_i is called an outgoing buffer for C_{i-1} and incoming one for C_i, $2 \leq i \leq K$. The loadlocks for C_1 can be thought of as just an incoming buffer. There are a number of processing steps (PSs) in each individual cluster tool. In this paper, each buffer connecting two adjacent tools is treated as a processing step with processing time being zero. Let $n[i]+1$ be the number of steps for processing a type of wafers in C_i, including the incoming and outgoing steps. These steps are denoted as PS_{i0}, PS_{i1}, …, and $PS_{i(n[i])}$ with PS_{i0} and $PS_{i(b[i])}$ being the incoming and outgoing steps, respectively. We use R_i to denote the robot in C_i. Then, the routing of a wafer in a K-cluster tool is as follows: $PS_{10} \rightarrow PS_{11} \rightarrow \ldots \rightarrow PS_{1(b[1])}$ $(PS_{20}) \rightarrow PS_{21} \rightarrow \ldots \rightarrow PS_{2(b[2])}$ $(PS_{30}) \rightarrow \ldots \rightarrow PS_{(K-1)(b[K-1])}$ $(PS_{K0}) \rightarrow PS_{K1} \rightarrow \ldots \rightarrow PS_{K(n[K])} \rightarrow \ldots \rightarrow PS_{K0}$ $(PS_{(K-1)(b[K-1])}) \rightarrow PS_{(K-1)(b[K-1]+1)} \rightarrow \ldots \rightarrow PS_{30}$ $(PS_{2(b[2])}) \rightarrow PS_{2(b[2]+1)}$ $\rightarrow \ldots \rightarrow PS_{20}$ $(PS_{1(b[1])}) \rightarrow \ldots \rightarrow PS_{1(n[1])} \rightarrow PS_{10}$. At most of time, a multi-cluster tool operates in a steady state. Under such state, if the backward scheduling is applied, robot R_i for C_i operates as follows. Robot R_i moves to $PS_{i(n[i])} \rightarrow$ unloads a processed wafer from there $\rightarrow$ moves to $PS_{i0} \rightarrow$ drops the wafer there $\rightarrow$ moves to $PS_{i(n[i]-1)} \rightarrow$ unloads a processed wafer from there $\rightarrow$ moves to $PS_{i(n[i])} \rightarrow$ drops the wafer there $\rightarrow \ldots \rightarrow$ moves to $PS_{i1} \rightarrow$ unloads a processed wafer from there $\rightarrow$ moves to $PS_{i2} \rightarrow$ drops the wafer there $\rightarrow$ moves to $PS_{i0} \rightarrow$ unloads a raw wafer from there $\rightarrow$ moves to $PS_{i1} \rightarrow$ drops the wafer there $\rightarrow$ moves to $PS_{i(n[i])}$. In this way, a robot cycle is completed.

2.2 ROPN for Wafer Flows

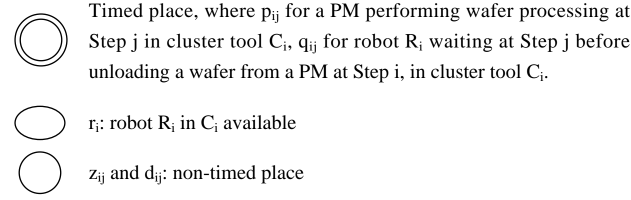
In this paper, the resource-oriented PN (ROPN) developed in [19; 23-26, 32] is used to model a K-cluster tool. It is a kind of finite capacity PN defined as PN = (P, T, I, O, M, K), where P is a finite set of places; T is a finite set of transitions, $P \cup T \neq \emptyset$, $P \cap T = \emptyset$; $I : P \times T \rightarrow \mathbf{N} = \{0, 1, 2, \ldots\}$ is an input function; O: $P \times T \rightarrow N$ is an output function; M: $P \rightarrow \mathbf{N}$ is a marking representing the numbers of tokens in places with M_0 being the initial marking; and K: $P \rightarrow \{1, 2, \ldots\}$ is a capacity function where $K(p)$ represents the number of tokens that p can hold at a time. The *preset* of transition t is the set of all input places to t, i.e. ${}^{\bullet}t = \{p$: $p \in P$ and $I(p, t) > 0\}$. Its *postset* is the set of all output places from t, i.e. $t^{\bullet} = \{p$: $p \in P$ and $O(p, t) > 0\}$. Similarly, p's *preset* ${}^{\bullet}p = \{t \in T$: $O(p, t) > 0\}$ and *postset* $p^{\bullet} = \{t \in T$: $I(p, t) > 0\}$. The transition enabling and firing rules can be found in [35, 25].

As pointed out in [5], to search for an optimal steady state schedule of a multi-cluster tool, an appropriate initial state should be given to the model. In operating a multi-cluster tool, there are three major states: the initial transient, steady, and final transient states. When a multi-cluster tool is turned on, it enters an initial transient state until a steady state is reached. Then, the system operates in the steady state. When the system needs to stop its operation, it enters a final transient state. It is extremely useful to develop a model that can describe the dynamic behavior for all three states. Also, the wafer types to be processed by a multi-cluster tool may be changed from time to time; and the wafer flow patterns are changed accordingly. A model just for the processing of a single wafer type is not reusable. Thus, it is desired that a model is independent of wafer flow patterns and can describe the behavior of all three states. Such a PN model is obtained for single-cluster tools in [20-22, 27-31] by using the ROPN modeling method proposed in [19, 23]. This work extends the work in [20-22, 27-31] to develop a PN model that is wafer flow pattern independent and can describe the dynamic behavior of all three states for multi-cluster tools, which is never done to the best knowledge of the authors.

As pointed out above, the key effort to schedule a multi-cluster tool is to coordinate the activities of multiple robots in loading and unloading the buffers. Thus, to model a multi-cluster tool for the scheduling purpose, we must model the discrete-event behavior of individual cluster tools and buffers. We present the PN model for an individual cluster tool first, and then develop it for a buffer. The robot in each tool is scheduled by a backward sequence. Multiple PMs can be configured for a step in a tool. For simplicity of presentation, without loss of generality, we assume that there is only one PM at a step except Step 0 in C_1 for the loadlocks that has infinite capacity. Based on this assumption, the behavior of Step j in C_i is modeled as follows.

As a fabrication resource, the PM for Step j in C_i with $i \neq 1$ and $j \neq 0$ is modeled by timed place p_{ij} with $K(p_{ij}) = 1$. We use place p_{10} to model the loadlocks in C_1 with $K(p_{10}) = \infty$. Notice that the only difference between p_{10} and p_{ij} with $i \neq 1$ and $j \neq 0$ is their capacity. Thus, we do not need to distinguish them when the wafer flow is modeled. Places z_{ij} and d_{ij} are used to model that robot R_i in C_i holds a wafer (token) for loading into p_{ij} and moving to Step j + 1 (or Step 0 if $j = n[i]$), respectively. According to [20, 27] for scheduling single-cluster tools, the robot waiting should also play an important role in developing a scheduling method for a multi-cluster tool in this paper. To model it, place q_{ij} is used to model that robot R_i in C_i waits at Step j for unloading a completed wafer in PM$_{ij}$. Then, transitions t_{ij} and u_{ij} are used to model that the robot R_i loads a wafer into PM$_{ij}$ and unloads a completed wafer from PM$_{ij}$, respectively. The icons used for the places and transitions of the model are shown in Fig. 2. By adding arcs (z_{ij}, t_{ij}), (t_{ij}, p_{ij}), (p_{ij}, u_{ij}), (q_{ij}, u_{ij}), and (u_{ij}, d_{ij}), we complete the modeling of Step j in C_i as shown in Fig. 3.

PLACES:



Timed place, where p_{ij} for a PM performing wafer processing at Step j in cluster tool C_i, q_{ij} for robot R_i waiting at Step j before unloading a wafer from a PM at Step i, in cluster tool C_i.

r_i: robot R_i in C_i available

z_{ij} and d_{ij}: non-timed place

TRANSITIONS:

Timed transition. t_{ij} for robot R_i loading a wafer into a PM at Step j in C_i; u_{ij} for robot R_i unloading a wafer from a PM at Step j in C_i; x_{i0} for robot R_i moving from Step 0 to 1, x_{ij}, j = 1, ⋯ n[i]-1, for robot R_i moving from Step j to j+1 in C_i, $x_{i(n[i])}$ for robot R_i moving from Step n[i] to 0 in C_i; y_{i0} for Robot R_i moving from Step 2 to 1 in C_i, y_{ij} for robot R_i Moving from Step j+2 to j, j = 1, ⋯, n[i]-2, $y_{i(n[i]-1)}$ for robot R_i moving from Step 0 to n[i]-1, and $y_{i(n[i])}$ for robot R_i moving from Step 1 to n[i] in C_i.

Fig. 2. Icons used in the PN model

With the model for a processing step, we can then model C_i as follows. Place r_i is used to model R_i with $K(r_i) = 1$, meaning that the robot has only one arm and can hold one wafer at a time. Transition y_{ij} is used to connect r_i and q_{ij} with an arc from r_i to y_{ij} and another from y_{ij} to q_{ij}. Finally, transition x_{ij}, j = 0, 1, 2, …, and $n[i]$-1, is added between places d_{ij} and $z_{i(j+1)}$ with an arc from d_{ij} to x_{ij} and another from x_{ij} to $z_{i(j+1)}$; $x_{i(n[i])}$ is added between $d_{i(n[i])}$ and z_{i0} with an arc from $d_{i(n[i])}$ to $x_{i(n[i])}$ and another from $x_{i(n[i])}$ to z_{i0}. In this way, the modeling of C_i is completed and it is shown in Fig. 3.

To effectively operate a multi-cluster tool is to effectively coordinate its multiple robots. Because the buffers are the shared resources among tools/robots, the key is effectively to schedule the activities that involve them. Let BM shared by C_i and C_{i+1} be PM$_{i(b[i])}$ as the outgoing buffer for C_i and PM$_{(i+1)0}$ as the incoming buffer or virtual loadlock for C_{i+1}. It is treated as a processing step for both C_i and C_{i+1} called Step $b[i]$ for C_i and Step 0 for C_{i+1}, respectively. Places $p_{i(b[i])}$ and $p_{(i+1)0}$ with $p_{i(b[i])} = p_{(i+1)0}$ are used to model this BM. Then, with the PN model for a

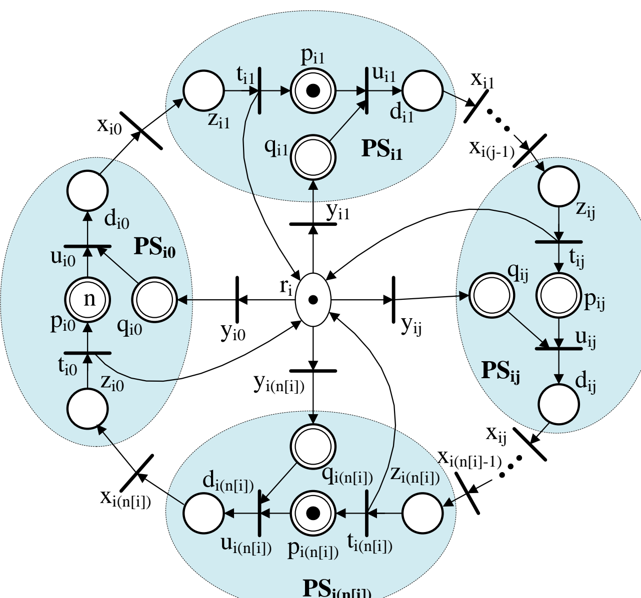


Fig. 3. The PN model for an individual cluster tool C_i

processing step, Step $b[i]$ for C_i is modeled by $\{p_{i(b[i])}$, $q_{i(b[i])}$, $z_{i(b[i])}$, $d_{i(b[i])}$, $t_{i(b[i])}$, $u_{i(b[i])}\}$ and Step 0 for C_{i+1} by $\{p_{(i+1)0}$, $q_{(i+1)0}$, $z_{(i+1)0}$, $d_{(i+1)0}$, $t_{(i+1)0}$, $u_{(i+1)0}\}$ as shown in Fig. 4. Notice that $p_{i(b[i])}$ and $p_{(i+1)0}$ are for the same BM. Thereafter, when we refer to Step $b[i]$ in C_i, we use $p_{i(b[i])}$, while for Step 0 in C_{i+1} we use $p_{(i+1)0}$. It should be pointed out that Step 0 in C_1 (the loadlocks) is not shared by any other cluster tool and there is no outgoing buffer in C_K.

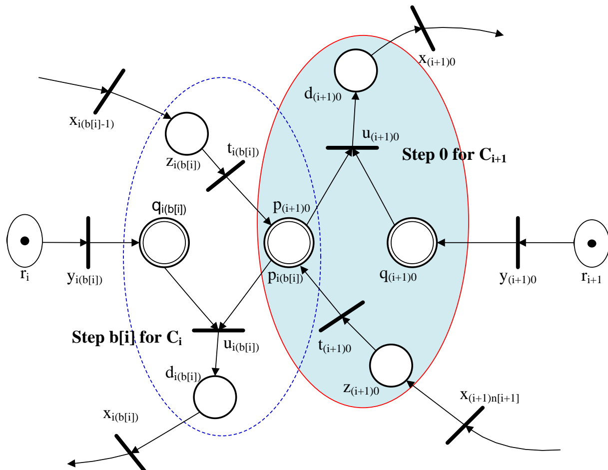


Fig. 4. The PN model for a buffer

Up to now, we present the structure of a PN model for the system, but not the initial marking M_0. For a process-dominant multi-cluster tool in its steady state, if every PM_{ij} has a wafer being processed with $j \neq 0$, the system reaches its maximal throughput. Thus, we let

(1) $M_0(p_{10}) = n$, $M_0(p_{1(b[1])}) = 0$, and $M_0(p_{1j}) = 1$ with $j \notin \{0, b[i]\}$;

(2) For $K > i \geq 2$, $M_0(p_{ij}) = 1$ with $j \notin \{0, b[i]\}$ and $M_0(p_{ij}) = 0$ with $j \in \{0, b[i]\}$;

(3) $M_0(p_{K0}) = 0$, $M_0(p_{Kj}) = 1$ with $j \notin 0$; and

(4) For any $i \in \mathrm{N}_K$ and j, $M_0(q_{ij}) = M_0(z_{ij}) = M_0(d_{ij}) = 0$, and $M_0(r_i) = 1$ as the initial marking for the PN model.

Observing the PN model for a multi-cluster tool developed above, at M_0, any transition of y_{ij} is both process and resource-enabled according to the transition enabling rule. Assume that PM_{11} is not a BM and, at marking M_0, we fire y_{10}, then u_{10}, followed by x_{10} such that a token moves into z_{11}. At this marking, t_{11} is process-enabled, but not resource-enabled for $M_0(p_{11}) = K(p_{11}) = 1$, leading to a dead marking. To solve this problem, we make the PN model a controlled PN. A PN is said to be controlled if at least one of its transitions can be controlled. A transition t is said to be controlled if its firing is determined by a control policy when t is both process and resource-enabled according to the enabling rule. Thus, a transition in a controlled PN is enabled if it is process, resource, and control-enabled. Here, all the transitions y_{ij} are controlled transitions. The control policy is defined as follows.

Definition 2.1: At marking M, transition y_{ij}, $i \in \mathrm{N}_K$ and $j \in \mathbf{\Omega}_{n[i]\text{-}1}$, is said to be control-enabled if $M(p_{i(j+1)}) = 0$; $y_{1(n[i])}$ is so if $M(p_{1(n[i])}) = 1$; and $y_{i(n[i])}$, $i \in \mathbf{N}_K-\{1\}$, is so if $M(p_{i0}) = 0$.

Thereafter, we assume that the PN model is controlled by the control policy given in Definition 2.1 .

Observe the PN model for a buffer shown in Fig. 4. Place $p_{i(b[i])}$ ($p_{(i+1)0}$) has two output transitions $u_{i(b[i])}$ and $u_{(i+1)0}$. Hence, there is a conflict when there is a token in $p_{i(b[i])}$ ($p_{(i+1)0}$). When a token enters $p_{i(b[i])}$ by firing $t_{i(b[i])}$ for Step $b[i]$ in C_i, it should enable $u_{(i+1)0}$ for Step 0 in C_{i+1}. However, when a token enters $p_{(i+1)0}$ by firing $t_{(i+1)0}$ for Step 0 in C_{i+1}, it should enable $u_{i(b[i])}$ for Step $b[i]$ in C_i. To avoid such a conflict, colors are introduced into the model. We first define the color for a transition.

Definition 2.2: Define the color of transition t_i as $C(t_i) = \{c_i\}$.

By Definition 2.2, the colors for $u_{i(b[i])}$ and $u_{(i+1)0}$ are $c_{i(b[i])}$ and $c_{(i+1)0}$, respectively. With Definition 2.2, we can define the color for a token as follows.

Definition 2.3: If a token in $p \in {}^{\bullet}t_i$ enables t_i, then it has the same color c_i as t_i.

For example, when a token enters $p_{i(b[i])}$ by firing $t_{i(b[i])}$ for Step $b[i]$ in C_i, it has color $c_{(i+1)0}$, while it enters $p_{(i+1)0}$ by firing $t_{(i+1)0}$ for Step 0 in C_{i+1}, it has color $c_{i(b[i])}$ for Step $b[i]$ in C_i. In this way, the transition enabling and firing rules for colored PN can be applied and the conflicts are resolved.

The proposed PN model can also describe the behavior of an initial and final transient process by using a special type of tokens, called W_0. For example, we initially put all W_0-tokens in p_{ij}'s to form M_0 such that all tokens in the system, except that in p_{10}, are W_0 tokens. With M_0 and the transition enabling and firing rules, the PN model can be run by just using a backward strategy as follows to reach the steady state. Without loss of generality, we assume that, at M_0, robot R_i for C_i is at Step $n[i]$, or $M_0(q_{i(n[i])}) = 1$.

Then, for C_1, the following transition firing sequence is executed: firing transition $u_{1(n[1])} \rightarrow x_{1(n[1])} \rightarrow t_{10} \rightarrow y_{1(n[1]\text{-}1)}$

$\rightarrow u_{1(n[1]-1)} \rightarrow x_{1(n[1]-1)} \rightarrow t_{1(n[1])} \rightarrow \ldots \rightarrow y_{1(b[1])}$.

At the same time, for C_i, $2 \le i \le K-1$, the following sequence is executed: $u_{i(n[i])} \rightarrow x_{i(n[i])} \rightarrow t_{i0} \rightarrow y_{i(n[i]-1)} \rightarrow u_{i(n[i]-1)} \rightarrow x_{i(n[i]-1)} \rightarrow t_{i(n[i])} \rightarrow \ldots \rightarrow y_{i(b[i])}$.

For C_K, the following sequence is executed: $u_{K(n[K])} \rightarrow x_{K(n[K])} \rightarrow t_{K0} \rightarrow y_{K(n[K]-1)} \rightarrow u_{K(n[K]-1)} \rightarrow x_{K(n[K]-1)} \rightarrow t_{K(n[K])} \rightarrow \ldots \rightarrow y_{K0}$.

Then, for C_1, after t_{20} is fired, the following sequence is executed: $u_{1(b[1])} \rightarrow x_{1(b[1])} \rightarrow t_{1(b[1]+1)} \rightarrow y_{1(b[1]-1)} \rightarrow \ldots \rightarrow y_{10} \rightarrow u_{10}$ (unloading a real wafer W_1) $\rightarrow x_{10} \rightarrow t_{11}$ (loading W_1 into Step 1) $\rightarrow y_{1(n[1])}$.

For C_i, $2 \le i \le K-1$, after $t_{(i+1)0}$ and $t_{(i-1)(b[i-1])}$ are fired, the following sequence is executed: $u_{i(b[i])} \rightarrow x_{i(b[i])} \rightarrow t_{i(b[i]+1)} \rightarrow y_{i(b[i]-1)} \rightarrow \ldots \rightarrow y_{i0} \rightarrow u_{i0} \rightarrow x_{i0} \rightarrow t_{i1} \rightarrow y_{i(n[i])}$.

For C_K, after $t_{(K-1)(b[K-1])}$ is fired, the following sequence is executed: $u_{K0} \rightarrow x_{K0} \rightarrow t_{K1} \rightarrow y_{K(n[K])}$. In this way, a marking M that is equivalent to M_0 is reached. Hence, the above process can be repeated until that all W_0 tokens come into P_{10}. At this time, the steady state is reached. When the system should be stopped, we change all tokens in p_{10} into W_0-tokens. Every time u_{10} fires, we unload a W_0 token from p_{10}. When all tokens in the model are type W_0, the final transient process ends. In this way, the model can describe all three states without changing the net structure.

2.3 Modeling Activity Time

To schedule a multi-cluster tool is to determine the start time of every activity. Thus, the model should describe the temporal aspect of the system. In the PN model, all transitions are for robot tasks that take time and some of the places, such as p_{ij} and q_{ij}, are timed ones. Hence, time is associated with both transitions and places in the model. If time ζ is associated with transition t, firing t takes ζ time units. If it is associated with p, a token must stay in p for at least ζ time units before it can enable p's output transition. As done in [1], we assume that the time taken by robot R_i in C_i to load/unload a wafer into/from a PM is same, as denoted by λ_i. Also, the time taken by robot R_i in C_i to move from a PM to another is same, as denoted by μ_i, regardless of whether the robot carries a wafer or not. The time taken for processing a wafer in a PM at Step j for C_i is α_{ij}. If Step j for C_i is a buffering step, including the loadlocks, $\alpha_{ij} = 0$, and otherwise it is greater than zero. We use ω_{ij} to denote R_i's waiting time in q_{ij} and τ_{ij} the wafer sojourn time in a PM at Step j for C_i.

3. INDIVIDUAL CLUSTER TOOL SCHEDULING

Based on its PN model we analyze the time taken to complete a wafer at Step j in cluster tool C_i with robot R_i's waiting time being considered. Without loss of generality, we assume that for any C_i we have $n[i] \ge 2$. It follows from [20] that the time taken for processing a wafer at Step j in C_i is

$$\xi_{ij} = \alpha_{ij} + 4\lambda_i + 3\mu_i + \omega_{i(j-1)},\ j \in \mathbf{N}_{n[i]} \quad (3.1)$$

For Step 0, $\alpha_{i0} = 0$ and we have

$$\xi_{i0} = 4\lambda_i + 3\mu_i + \omega_{i(n[i])} \quad (3.2)$$

It follows from (3.1) and (3.2) that robot waiting has effect on the processing time for completing a wafer in a PM. By removing the robot waiting time in (3.1) and (3.2), we have

$$\eta_{ij} = \alpha_{ij} + 4\lambda_i + 3\mu_i,\ j \in \mathrm{N}_{n[i]} \quad (3.3)$$

$$\text{and} \quad \eta_{i0} = 4\lambda_i + 3\mu_I \quad (3.4)$$

Expressions (3.3) and (3.4) present the shortest time needed for processing a wafer at Step j in C_i. If a wafer stays in the PM at Step j for more than the required processing time, or $\tau_{ij} \ge \alpha_{ij}$, it is still a feasible. Thus, by replacing α_{ij} by $\tau_{ij} \ge \alpha_{ij}$, we have

$$\theta_{ij} = \tau_{ij} + 4\lambda_i + 3\mu_i + \omega_{i(j-1)},\ j \in \mathrm{N}_{n[i]} \quad (3.5)$$

$$\text{and} \quad \theta_{i0} = \tau_{i0} + 4\lambda_i + 3\mu_i + \omega_{i(n[i])} \quad (3.6)$$

Observe the PN shown in Fig. 4 for buffering Step $b[i]$ for C_i. When wafer W_k is loaded into $p_{i(b[i])}$ by firing $t_{i(b[i])}$, W_k can be unloaded immediately by firing $u_{(i+1)0}$. Similarly, for Step 0 in C_{i+1}, a wafer loaded by firing $t_{(i+1)0}$ can be unloaded immediately by firing $u_{i(b[i])}$. It means that the real wafer sojourn time in a buffering step depends not only on the schedule of cluster tool C_i but also that of C_{i+1}. We define the virtual wafer sojourn time for a buffering step. For buffering Step j in C_i, let υ_{j1} and υ_{j2} be the time points when t_{ij} and u_{ij} fire for their k-th time, respectively. Next, define $\tau_{ij} = \upsilon_{j2} - \upsilon_{j1}$ as the virtual wafer sojourn time at buffering Step $j = b[i]$ or 0. From the viewpoint of cluster tool C_i, it is equivalent that there is a wafer staying there for τ_{ij} time units. Thus, thereafter, when we mention wafer sojourn time in Step j for C_i, it means the real wafer sojourn time if it is a processing step, or the virtual wafer sojourn time if it is a buffering step.

In the PN model shown in Fig. 3, for cluster tool C_i, assume that marking M is reached such that $M(p_{ij}) = 1$, $j \in \mathbf{\Omega}_{n[i]} - \{1\}$, $M(q_{i0}) = 1$, and $M(p_{i1}) = 0$. Then, to complete a robot cycle in C_i, a transition firing sequence in the backward schedule should be executed. By using the results in [20], we have the cycle time of R_i

$$\psi_i = 2(n[i] + 1)(\lambda_i + \mu_i) + \sum_{j=0}^{n[i]} \omega_{ij} = \psi_{i1} + \psi_{i2},\ i \in \mathbf{N}_K \quad (3.7)$$

where $\psi_{i1} = 2(n[i] + 1)(\lambda_i + \mu_i)$ is the robot cycle time without waiting and is a constant, while $\psi_{i2} = \sum_{j=0}^{n[i]} \omega_{ij}$ is the robot waiting time in a cycle.

Let $\Pi_i = \max\{\eta_{i0}, \eta_{i1}, \ldots, \eta_{i(n[i])}, \psi_{i1}\}$. If $\Pi_i = \max\{\eta_{i0}, \eta_{i1}, \ldots, \eta_{i(n[i])}\}$, cluster C_i is process-bound, otherwise it is transport-bound. For each individual tool C_i, it is a serial manufacturing process. Thus, for a one-wafer schedule in the steady state, the productivity for each step must be same, or the time to complete a wafer for every step in C_i must be same. This implies that C_i should be scheduled such that

$$\Theta_i = \theta_{i0} = \theta_{i1} = \ldots = \theta_{i(n[i])} \text{ and } \tau_{ij} \ge \alpha_{ij},\ i \in \mathbf{N}_K,\ j \in \mathbf{\Omega}_{n[i]} \quad (3.8)$$

By (3.8), during Θ_i, one wafer is completed in C_i, or Θ_i

is the cycle time for the wafer processing process. Note that a wafer is completed during ψ_i. Thus, in steady state, C_i should be scheduled such that $\psi_i = \Theta_i$. In this case, $\psi_{i2} = \Theta_i - \psi_{i1}$ is the R_i's idle time for a cycle. In other words, the sum of robot R_i's waiting time in a cycle should be equal to its idle time.

Assume that $\Pi_i = \max\{\eta_{i0}, \eta_{i1}, \ldots, \eta_{i(n[i])}\}$. Then, if $\Theta_i < \Pi_i$, it follows from (3.3) - (3.6) that there is at least $j \in \mathbf{\Omega}_{n[i]}$ such that $\tau_{ij} < \alpha_{ij}$. In this case, no infeasible schedule can be found. When $\Pi_i = \psi_{i1}$ and $\Theta_i < \Pi_i$, R_i is not fast enough to complete a wafer during Θ_i, leading to an infeasible schedule. However, if $\Theta_i \geq \Pi_i$, $\tau_{ij} \geq \alpha_{ij}$ can be satisfied by appropriately setting ω_{ij}'s and, at the same time, R_i is fast enough to complete a wafer in a cycle. Thus, to schedule C_i, $i \in \mathbf{N}_K$, is to determine $\Theta_i \geq \Pi_i$ and ω_{ij}'s such that (3.8), $\psi_{i2} = \Theta_i - \psi_{i1}$, and $\Theta_i = \psi_i$ are all satisfied. In this way, given Θ_i, we parameterize the schedule of individual cluster tools by ω_{ij}'s.

4. Individual Cluster Tool Scheduling

Based on the result developed in the last section, this section presents the scheduling method for a K-cluster tool composed of K single-cluster tools. Before doing so, we first establish the optimality conditions.

4.1 Optimality Analysis

Let $\Pi = \max\{\Pi_1, \Pi_2, \ldots, \Pi_K\}$ and Θ be the scheduled cycle time of a K-cluster tool. Further, let $\Pi = \Pi_h$, or the FP of C_h is the largest one among the single-cluster tools that form the K-cluster tool. Without loss of generality, we assume that $2 \leq h \leq K-1$. The results obtained thereafter can be easily extended to the cases with $h = 1$ and $h = K$. By assumption, C_h is process-bound and such a K-cluster tool is called process-dominant multi-cluster tool. In cluster tool C_i, there are $n[i]+1$ steps, including two buffering steps in C_i ($i \neq K$) numbered as 0 and $b[i]$. For C_K, there is only one buffering Step 0. Thus, for $i \neq K$, there are $n[i] - 1$ processing steps, or $n[i] - 1$ wafers can be concurrently processed. For $i = K$, there are $n[i] + 1$ steps and one buffering step, or $n[i]$ wafers can be concurrently processed. According to [1, 2], only when cluster tool C_i is transport-bound, it is possible to shorten Π_i by reducing the number of wafers that are concurrently processed. This implies that backward scheduling is optimal for C_h with $\Pi_h = \Pi$ being the lower bound of cycle time and, under such a scheduling strategy, there are $n[h] - 1$ wafers that are being concurrently processed. Then, we have the following result.

Proposition 4.1: Under its steady state, a process-dominant K-cluster tool has the lower bound of cycle time of C_h, or

$$\Theta \geq \Pi = \Pi_h \tag{4.1}$$

With Proposition 4.1, we call C_h the bottleneck in the K-cluster tool. Proposition 4.1 means that if a schedule can be found for a process-dominant K-cluster tool such that $\Theta = \Pi$, it must be optimal in terms of its cycle time.

Proposition 4.2: If cluster tool C_i of a K-cluster tool is scheduled such that its cycle time is Θ_i, the cycle times of C_{i-1} and C_{i+1} must be greater than or equal to Θ_i, or

$$\Theta_{i-1} \geq \Theta_i \tag{4.2}$$

$$\text{and} \quad \Theta_{i+1} \geq \Theta_i \tag{4.3}$$

Proof: Consider the PN model for the buffering step shared by C_{i-1} and C_i shown in Fig. 4. Assume that, at marking M, t_{i0} fires and a token is put into p_{i0} ($p_{(i-1)(b[i-1])}$). By definition, this token enables $u_{(i-1)(b[i-1])}$. Further, assume that, at the end of t_{i0}'s firing, a token in $q_{(i-1)(b[i-1])}$ is available. Thus, after firing t_{i0}, $u_{(i-1)(b[i-1])}$ fires immediately. Then, although a token may enter $q_{(i-1)(b[i-1])}$ again at time $\tau < \Theta_i$, a token can be put into p_{i0} ($p_{(i-1)(b[i-1])}$) by firing t_{i0} again only Θ_i time units later. Thus, the token in $q_{(i-1)(b[i-1])}$ has to wait for the arrival of the token in p_{i0} ($p_{(i-1)(b[i-1])}$) before $u_{(i-1)(b[i-1])}$ can fire again. This implies that $u_{(i-1)(b[i-1])}$ can fire again at least Θ_i time units later. Because the process for any individual cluster tool is a serial one, (4.2) must hold. Similarly, we can show that (4.3) holds. ■

It follows from (4.2) and (4.3) that, for C_i and C_{i+1}, we have $\Theta_{i+1} \geq \Theta_i$ and $\Theta_i \geq \Theta_{i+1}$. In other words, $\Theta_i = \Theta_{i+1}$ must hold. Thus, we have the following result.

Proposition 4.3: A K-cluster tool should be scheduled such that all individual cluster tools have the same cycle time and it is equal to the cycle time of the K-cluster tool, or

$$\Theta_1 = \Theta_2 = \ldots = \Theta_K = \Theta \tag{4.4}$$

A K-cluster tool can be seen as a flow line with a single-cluster tool being an operator. Thus, when a K-cluster tool is scheduled such that (4.4) holds, its operation must be paced. It follows form Propositions 4.1 and 4.3 that we have the following result immediately for finding an optimal one-wafer schedule.

Theorem 4.1: To find an optimal one-wafer periodic schedule for a process-dominant K-cluster tool, the system should be scheduled such that

$$\Theta_1 = \Theta_2 = \ldots = \Theta_K = \Theta = \Pi = \Pi_h \tag{4.5}$$

By Theorem 4.1, it means that each individual tool C_i in a K-cluster tool should be scheduled such that $\Theta_i \geq \Pi_i$. If C_i is process-bound, we have $\Theta_i \geq \Pi_i = \max\{\eta_{i0}, \eta_{i1}, \ldots, \eta_{i(n[i])}\}$ that can be achieved by backward scheduling with $n[i] - 1$, $i \neq K$ ($n[i]$ if $i = K$) wafers being concurrently processed. In fact, according to the method for individual cluster tool scheduling presented in the last section, this can be done by appropriately determining the robot waiting time ω_{ij}'s. If C_i is transport-bound, we have $i \neq h$ and $\Theta_i > \Pi_i = \psi_{i1}$. This means that it is meaningless to shorten $\Pi_i = \psi_{i1}$ by reducing the number of wafers that are concurrently processed in C_i. In other words, for such a case, backward scheduling can still be applied without deteriorating the performance of the K-cluster tool. Hence, for a process-dominant K-cluster tool, every individual cluster tool can be scheduled by a backward strategy. Then, according to

the scheduling method for individual cluster tools, to optimally schedule a process-dominant K-cluster tool is to determine ω_{ij}'s for $i \in \mathbf{N}_K$ and $j \in \mathbf{\Omega}_{n[i]}$ such that its multiple robots can be optimally coordinated. We have the following result.

Theorem 4.2: Robots R_i and R_{i-1} can operate independently with the lower bound cycle time Π_h, if and only if, for C_i and C_{i-1}, $2 \le i \le K$, the following conditions are satisfied by determining ω_{ij}'s and $\omega_{(i-1)f}$'s:

1) $\theta_{ij} = \theta_{(i-1)f} = \Theta = \Pi_h$, $j \in \mathbf{\Omega}_{n[i]}$ and $f \in \mathbf{\Omega}_{n[i-1]}$; and
2) the virtual wafer sojourn time τ_{i0} is not less than $4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$.

Proof: To obtain an optimal one-wafer periodic schedule for a process-dominant K-cluster tool, (4.5) must be satisfied. Based on the method of individual cluster tool scheduling presented in the last section, to make $\Theta_i = \Theta = \Pi_h$, if and only if C_i is scheduled such that the time taken for completing a wafer at every Step j, $j \in \mathbf{\Omega}_{n[i]}$, is $\Theta_i = \Theta = \Pi_h$. Thus, condition 1) should hold. According to the method of individual cluster tool scheduling, such a schedule is determined by the values of ω_{ij}'s and $\omega_{(i-1)f}$'s. Next, we need to show the necessity and sufficiency of the condition 2).

Necessity. In C_i, it follows from condition 1) and the scheduling method for individual cluster tools that $\Theta = \Pi_h = \tau_{i0} + 4\lambda_i + 3\mu_i + \omega_{i(n[i])}$ holds. From the viewpoint of C_i, $\tau_{i0} = \Theta - (4\lambda_i + 3\mu_i + \omega_{i(n[i])})$. With the PN model for the buffering Step 0 in C_i, let ϕ_1 denote the time point when firing t_{i0}, i.e., loading a wafer into p_{i0}, ends, and ϕ_2 the time point when the first firing u_{i0}, i.e., unloading a wafer from p_{i0}, starts after firing t_{i0}. Then, it follows from the definition of virtual wafer sojourn time in a buffer and the schedule for C_i that we have $\tau_{i0} = \phi_2 - \phi_1$. Assume that, after firing t_{i0}, transition $u_{(i-1)(b[i-1])}$ in Step $b[i-1]$ for C_{i-1} fires immediately at ϕ_1 to unload the wafer loaded into $p_{(i-1)(b[i-1])}$ by firing t_{i0}. By starting from time point ϕ_1, it undergoes the following transition firing sequence for R_i in C_i: $\sigma_1 = \langle u_{(i-1)(b[i-1])} \rightarrow x_{(i-1)(b[i-1])} \rightarrow t_{(i-1)(b[i-1]+1)} \rightarrow y_{(i-1)(b[i-1]-1)} \rightarrow$ waiting in $q_{(i-1)(b[i-1]-1)}$ with $\omega_{(i-1)(b[i-1]-1)} \rightarrow u_{(i-1)(b[i-1]-1)} \rightarrow x_{(i-1)(b[i-1]-1)} \rightarrow t_{(i-1)(b[i-1])}\rangle$. Let ϕ_3 denote the time point when firing $t_{(i-1)(b[i-1])}$ ends. After its firing, a wafer is loaded into $p_{(i-1)(b[i-1])}$ (p_{i0}) at ϕ_3 to enable u_{i0}. This means that u_{i0} cannot start firing at ϕ_2, but ϕ_3. Thus, from the viewpoint of C_i, the time taken for completing a wafer at Step 0 is $\Theta' = (\phi_3 - \phi_1) + 4\lambda_i + 3\mu_i + \omega_{i(n[i])}$ but not $\Theta = \tau_{i0} + 4\lambda_i + 3\mu_i + \omega_{i(n[i])}$. Because the time taken by transition firing sequence σ_1 is $4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$, we have $\phi_3 - \phi_1 = 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$. Hence, assume that condition (2) is not satisfied, or $\phi_3 - \phi_1 = 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)} > \tau_{i0}$, then $\Theta' > \Theta$, or the cycle time of C_i is greater than Θ. This shows the necessity of condition 2).

Sufficiency. It follows from the individual cluster tool scheduling method given in the last section and condition 1) that every step in C_i is scheduled such that its cycle time is Θ. Obviously, if C_i does not share any step with other individual tools, the schedule can be executed for C_i with cycle time Θ. This implies that we need to examine the buffering steps in C_i only. If the schedule can be executed for the two buffering steps, such a schedule can be executed for C_i. With the PN model shown in Fig. 4, we examine buffering Step $b[i-1]$ in C_{i-1}. It is scheduled such that $\Theta = \Pi_h = \tau_{(i-1)(b[i-1])} + 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$. Let ϕ_4 denote the time point when firing t_{i0}, i.e., loading a wafer into $p_{(i-1)(b[i-1])}$, ends, and ϕ_5 the time point when the first firing u_{i0} for unloading a wafer from p_{i0} starts after firing t_{i0}. According to the schedule, we have $\phi_5 - \phi_4 = \tau_{i0}$. After firing t_{i0}, one can schedule C_{i-1} such that transition $u_{(i-1)(b[i-1])}$ fires immediately at ϕ_4 to unload the wafer loaded into $p_{(i-1)(b[i-1])}$ by firing t_{i0}. By starting from time point ϕ_4, it undergoes transition firing sequence σ_1. Let ϕ_6 denote the time point when firing t_{i0} ends. After firing t_{i0}, a wafer is loaded into $p_{(i-1)(b[i-1])}$ (p_{i0}) at ϕ_6 to enable $u_{(i-1)(b[i-1])}$. Notice that the time taken for executing σ_1 is $4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$ time units, or $\phi_6 - \phi_4 = 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$. By condition 2), we have $\phi_5 - \phi_4 = \tau_{i0} \ge 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)} = \phi_6 - \phi_4$. This implies that $\phi_6 \le \phi_5$, or a token is put into $p_{(i-1)(b[i-1])}$ (p_{i0}) before u_{i0} needs to fire. This implies that whenever u_{i0} is scheduled to fire, it is enabled. Thus, for buffering Step PS_{i0} the schedule is executable. If conditions 1) and 2) hold for C_i and C_{i+1}, by (3.5) and (3.6), we have $\tau_{(i+1)0} = \Theta - (4\lambda_{i+1} + 3\mu_{i+1} + \omega_{(i+1)(n[i+1])}) \ge 4\lambda_i + 3\mu_i + \omega_{i(b[i]-1)} = \Theta - \tau_{i(b[i])}$, thus, $\tau_{i(b[i])} \ge 4\lambda_{i+1} + 3\mu_{i+1} + \omega_{(i+1)(n[i+1])}$. Consider the buffering Step $PS_{i(b[i])}$ shared by C_i and C_{i+1}, one can similarly show that, the schedule is executable for Step $PS_{i(b[i])}$. Hence, for every C_i, the schedule is executable and the cycle time for the K-cluster tool is $\Theta = \Pi_h$, i.e., R_i and R_{i-1} can operate independently with the same cycle time Π_h. ■

The conditions given in Theorem 4.2 can be illustrated by Fig. 5 where τ_{i0} is a time window with $\phi_2 - \phi_1$ being its width. For C_i, R_i puts a wafer into the buffer at ϕ_1, and then at ϕ_2, R_i should take a wafer away from the buffer. For C_{i-1}, during the window $4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$, R_{i-1} unloads a wafer from and then puts a wafer into the buffer. Thus, there is a wafer to be unloaded by R_i at ϕ_2 only if $\tau_{i0} \ge 4\lambda_{i-1} + 3\mu_{i-1} + \omega_{(i-1)(b[i-1]-1)}$. In this way, R_i's activities are not affected by that of R_{i-1}'s. Theorem 4.2 presents the conditions under which the lower bound of cycle time can be obtained. However, it does not reveal how to verify the conditions, to be discussed next.

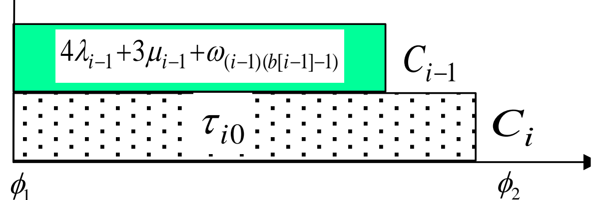


Fig. 5. Illustration of Theorem 4.2

4.2 Scheduling *K*-Cluster Tool

If conditions 1) and 2) in Theorem 4.2 hold, by (3.5) and (3.6), we have $\tau_{i(b[i])} \geq 4\lambda_{i+1} + 3\mu_{i+1} + \omega_{(i+1)(n[i+1])}$. It follows from the proof of Theorem 4.2 that, if every individual tool C_i in a K-cluster tool can be scheduled such that conditions 1) and 2) in Theorem 4.2 are satisfied, its multiple robots can be coordinated such that C_i, $i \in \mathbf{N}_K$, can operate as follows: whenever R_i needs to load a wafer into a buffering step as scheduled, the BM is idle, and also whenever it needs to unload a wafer from a buffering step as scheduled, there is a wafer available. In this way, we can think that each individual tool can operate independently without intervening each other. Notice that the conditions given in Theorem 4.2 are functions of ω_{ij}'s. Thus, the key question is how to determine ω_{ij}'s.

Let $\Theta = \Pi_h$ be the lower bound of cycle time for a K-cluster tool. For C_i, by examining the buffering Steps $PS_{i(b[i])}$ and PS_{i0}, we have $\Theta = \tau_{i(b[i])} + 4\lambda_i + 3\mu_i + \omega_{i(b[i]-1)}$ and $\Theta = \tau_{i0} + 4\lambda_i + 3\mu_i + \omega_{i(n[i])}$. Further, let $\phi_{i(b[i])} = 4\lambda_i + 3\mu_i + \omega_{i(b[i]-1)}$ and $\phi_{i0} = 4\lambda_i + 3\mu_i + \omega_{i(n[i])}$. To make the conditions in Theorem 4.2 satisfied, we need to determine ω_{ij}'s such that $\tau_{i(b[i])}$ and τ_{i0} are as large as possible, while $\phi_{i(b[i])}$ and ϕ_{i0} as small as possible. Notice that small $\omega_{i(b[i]-1)}$ results in large $\tau_{i(b[i])}$ and small $\phi_{i(b[i])}$ as well. Similarly, small $\omega_{i(n[i])}$ results in large τ_{i0} and small ϕ_{i0}. Furthermore, for C_i, we have $\Theta = \psi_i = 2(n[i] + 1)(\lambda_i + \mu_i) + \sum_{d=0}^{n[i]} \omega_{id} = \psi_{i1} + \psi_{i2}$, leading to R_i's idle time in a cycle $\psi_{i2} = \Theta - 2(n[i] + 1)(\lambda_i + \mu_i)$ that should be allocated to robot waiting time. A process-dominant K-cluster tool studied implies that $\Theta \geq \psi_{i1}$, or $\psi_{i2} \geq 0$. Hence, to optimally schedule a process-dominant K-cluster tool is to schedule C_i by allocating ψ_{i2} into ω_{ij}'s such that $\omega_{i(n[i])}$ and $\omega_{i(b[i]-1)}$ are as small as possible. By ψ_{i2}, $\Theta = \tau_{ij} + 4\lambda_i + 3\mu_i + \omega_{i(j-1)}$, and the scheduling method of individual cluster tools, it is easy to schedule C_i such that $\omega_{i(n[i])} + \omega_{i(b[i]-1)}$ is as small as possible. The key issue is how to determine the value of $\omega_{i(n[i])}$ and $\omega_{i(b[i]-1)}$ to meet the conditions in Theorem 4.2.

Notice that, for C_K, there is only one buffering step PS_{K0}. This implies that we need to make only $\omega_{K(n[K])}$ as small as possible and this can be done by using the method presented in the last section. With $\omega_{K(n[K])}$ determined, τ_{K0} is known. Then, for C_{K-1}, we can set the largest value to $\omega_{(K-1)(b[K-1]-1)}$ such that the conditions given in Theorem 4.2 for C_K and C_{K-1} are satisfied. Then, we can make $\omega_{(K-1)(n[K-1])}$ as small as possible. In this way, we can schedule a process-dominant K-cluster tool by scheduling the individual cluster tool one by one as presented by Algorithm 4.1 below. With Algorithm 4.1, its output Γ represents whether a one-wafer cyclic schedule with cycle time Π_h is found or not.

Algorithm 4.1: Schedule a process-dominant K-cluster by determining ω_{ij}'s.
Input: α_{ij}, λ_i, μ_i, $(i \in \mathrm{N}_k, j \in \Omega_{n[i]})$
Output: ω_{ij}, Γ, Q, $(i \in \mathrm{N}_k, j \in \Omega_{n[i]})$

i) Let $\Theta \leftarrow \max_{1 \leq i \leq K}(\Pi_i)$, $\Gamma \leftarrow 1$, $Q \leftarrow 0$
ii) Determine ω_{Kj}, $j \in \mathbf{\Omega}_{n[K]}$ for R_K as
  1) **For** $j \leftarrow 0$ to $n[K] - 1$ **do**
  2) $\omega_{Kj} \leftarrow \min\{\Theta - (4\lambda_K + 3\mu_K + \alpha_{K(j+1)}), \Theta - 2(n[K] + 1)(\lambda_K + \mu_K) - \sum_{m=0}^{j-1} \omega_{Km}\}$
  3) **EndFor**
  4) $\omega_{K(n[K])} \leftarrow \Theta - 2(n[K] + 1)(\lambda_K + \mu_K) - \sum_{m=0}^{n[K]-1} \omega_{Km}$
  5) $\tau_{K0} \leftarrow \Theta - (4\lambda_K + 3\mu_K + \omega_{K(n[K])})$
iii) If $K > 2$, determine ω_{ij} for R_i, $1 \leq i \leq K-1$, $j \in \mathbf{\Omega}_{n[i]}$ as.
  1) $i \leftarrow K - 1$
  2) **While** $i \geq 1$ **and** $\tau_{(i+1)0} \geq (4\lambda_i + 3\mu_i)$ **do**
  3) $\omega_{i(b[i]-1)} \leftarrow \min\{\tau_{(i+1)0} - (4\lambda_i + 3\mu_i), \Theta - 2(n[i] + 1)(\lambda_i + \mu_i)\}$
  4) **For** $j \leftarrow 0$ to $n[i]$ **and** $j \neq b[i] - 1$ **do**
  5) $\omega_{ij} \leftarrow \min\{\Theta - (4\lambda_i + 3\mu_i + \alpha_{i(j+1)}), \Theta - 2(n[i] + 1)(\lambda_i + \mu_i) - \sum_{m=0}^{j-1} \omega_{im}\}$
  6) **EndFor**
  7) $\tau_{i0} \leftarrow \Theta - (4\lambda_i + 3\mu_i + \omega_{i(n[i])})$
  8) $i \leftarrow i - 1$
  9) **EndWhile**
iv) Decide whether reaching the lower bound.
  1) **If** $i > 0$ **Then** $\Gamma \leftarrow 0$, $Q \leftarrow i$
  2) **EndIf**

By Algorithm 4.1, if it returns $\Gamma = 1$, for C_i, $1 \leq i \leq K - 1$, $\omega_{i(n[i])}$ and $\omega_{i(b[i]-1)}$ are determined such that the conditions in Theorem 4.2 are satisfied. By Algorithm 4.1, if it returns $\Gamma = 0$, its output Q represents that the conditions given in Theorem 4.2 for C_Q and C_{Q+1} are violated, i.e., $\tau_{(Q+1)0} \geq (4\lambda_Q + 3\mu_Q)$ does not hold. Thus, if it returns $\Gamma = 1$, a solution is obtained such that the conditions in Theorem 4.2 are satisfied. Thus, we have the following result immediately.

Theorem 4.3: If a one-wafer periodic schedule with the lower bound Π_h of cycle time for a process-dominant K-cluster tool is found by Algorithm 4.1 with its output $\Gamma = 1$, the schedule is optimal in terms of cycle time.

It follows from Algorithm 4.1 that the robots' waiting time determines if the condition given Theorem 4.3 can be satisfied. If it is not correctly set, such a schedule cannot be found even if it exists. If Algorithm 4.1 returns $\Gamma = 0$, it implies that $\tau_{(i+1)0} \geq (4\lambda_i + 3\mu_i)$ for $i \in \mathrm{N}_{K-1}$ does not hold even if $\tau_{(i+1)0}$ $(i \in \mathrm{N}_{K-1})$ is maximized by Algorithm 4.1, i.e., it is impossible to find the robots' appropriate waiting time to satisfy the conditions in Theorem 4.2. We have the following corollary.

Corollary 4.1: If Algorithm 4.1 is applied to a process-dominant K-cluster tool with its output $\Gamma = 0$, there is no 1-wafer cycle schedule reaching the lower bound of its cycle time.

Obviously, the computational complexity of Algorithm

4.1 is $O(K)$. Thus, it is extremely efficient and represents a significant progress in this research field.

4.3 Schedule Implementation

By Algorithm 4.1, when R_i in C_i loads a wafer into PS_{i0} by firing t_{i0}, $u_{(i-1)(b[i-1])}$ fires immediately to unload this wafer by robot R_{i-1} in C_{i-1}. This means that the multiple robots cannot act independently, but should act synchronously. Let ϕ_{K0}, the starting time point of firing t_{K0} in C_K, be the datum for the activities of R_K. For $i \in \mathbf{N}_{K-1}$, let $\phi_{i(b[i])}$, the starting time point for firing $u_{i(b[i])}$ in C_i, be the datum for the activities of R_i. For C_K, assume that the initial operation starts from firing t_{K0} to put a token into p_{K0}, which takes time λ_K. Let $\Delta_K = \lambda_K$. For $i \in \mathbf{N}_{K-1}$, between the firings of $u_{i(b[i])}$ and t_{i0}, the following transition firing sequence is executed:

$\sigma_i = \langle$firing $u_{i(b[i])}$ (with time λ_i) $\rightarrow x_{i(b[i])}$ $(\mu_i) \rightarrow t_{i(b[i]+1)}$ $(\lambda_i) \rightarrow y_{i(b[i]-1)}$ $(\mu_i) \rightarrow$ robot waiting in $q_{i(b[i]-1)}$ $(\omega_{i(b[i]-1)}) \rightarrow u_{i(b[i]-1)}$ $(\lambda_i) \rightarrow x_{i(b[i]-1)}$ $(\mu_i) \rightarrow t_{i(b[i])}$ $(\lambda_i) \rightarrow \ldots \rightarrow y_{i(n[i])}$ $(\mu_i) \rightarrow$ robot waiting in $q_{i(n[i])}$ $(\omega_{i(n[i])}) \rightarrow u_{i(n[i])}$ $(\lambda_i) \rightarrow x_{i(n[i])}$ $(\mu_i) \rightarrow t_{i0}$ $(\lambda_i)\rangle$. The time taken by σ_i is $\Delta_i = (b[i] - 1) \times (4\lambda_i + 3\mu_i) + 3\mu_i + 2\lambda_i + \omega_{i(n[i])} + \sum_{j=0}^{b[i]-1} \omega_{ij}$, where $b[i] > 1$. If $b[i] = 1$, $\Delta_i = \omega_{i0} + 6\lambda_i + 5\mu_i$. Thus, we have $\phi_{(K-1)(b[K-1])} = \phi_{K0} + \Delta_K$ and $\phi_{(i-1)(b[i-1])} = \phi_{i(b[i])} + \Delta_i$ for $2 \le i \le K - 1$. In other words, $u_{(K-1)(b[K-1])}$ should fire Δ_K time units later after the firing t_{K0}, and $u_{(i-1)(b[i-1])}$ should fire Δ_i time units later after the firing of $u_{i(b[i])}$ for $2 \le i \le K - 1$.

The above synchronization requirement can be implemented via the PN model as follows. Set the initial state M_0 by putting tokens W_0 representing virtual wafers into p_{ij}'s such that,

1) For $i \in \mathbf{N}_{K-2}$ ($K > 2$), let $M_0(p_{ij}) = 1$ for $j \ne b[i] + 1$, $M_0(p_{i(b[i]+1)}) = 0$;
2) $M_0(p_{(K-1)j}) = 1$ for $K > 1$, $j \ne b[K - 1]$ and $j \ne b[K - 1] + 1$, $M_0(p_{(K-1)(b[K-1]+1)}) = 0$; $M_0(p_{(K-1)(b[K-1])}) = M_0(p_{K0}) = 0$, $M_0(p_{Kj}) = 1$ for $j \ne 0$ and $j \ne n[K]$, and $M_0(p_{K(n[K])}) = 0$;
3) Further, assume that, for $2 \le i \le K$, a token in p_{i0} enables $u_{(i-1)(b[i-1])}$ and a token in $p_{(i-1)(b[i-1])}$ enables u_{i0}; and
4) For other places, $M_0(q_{ij}) = M_0(z_{ij}) = M_0(d_{ij}) = 0$ with $i \in \mathbf{N}_{K-1}$ and $j \in \Omega_{n[i]}$, $M_0(q_{Kj}) = M_0(d_{Kj}) = 0$, $M_0(z_{Kj}) = 0$ with $j \ne 0$, $M_0(z_{K0}) = 1$, $M_0(r_i) = 1$ with $i \in \mathbf{N}_{K-1}$, and $M_0(r_K) = 0$.

Thus, starting from the firing of t_{K0}, the PN runs as follows. After transition t_{K0} fires, a token goes into p_{K0} ($p_{(K-1)(b[K-1])}$), which results in the firing of $u_{(K-1)(b[K-1])}$. Thus, transition firing sequence σ_{K-1} is executed, where $u_{(K-1)(b[K-1])}$ fires at $\phi_{(K-1)(b[K-1])} = \phi_{K0} + \Delta_K = \lambda_K$. Next, σ_{K-2} is executed, where $u_{(K-2)(b[K-2])}$ fires at $\phi_{(K-2)(b[K-2])} = \phi_{(K-1)(b[K-1])} + \Delta_{K-1}$. This process is propagated from C_K to C_1 such that $\phi_{(K-1)(b[K-1])} = \phi_{K0} + \Delta_K$ and $\phi_{(i-1)(b[i-1])} = \phi_{i(b[i])} + \Delta_i$ for $2 \le i \le K - 1$, or the multiple robots in the system act synchronously according to the schedule. In this way, σ_1 is executed at $\phi_{1(b[1])} = \Delta_K + \Delta_{K-1} + \Delta_{K-2} + \ldots + \Delta_2$. After σ_1, every time when u_{10} fires a real token W (wafer) is removed from p_{10}. For each cycle performed in C_1, W_0-token goes into p_{10}. When all W_0 tokens go p_{10}, the steady state is reached. Then, the system operates under the given schedule. In this way, the schedule is implemented. It follows from the implementation that the one-wafer schedule obtained is a backward schedule. Thus, it is easy to implement.

5. Illustrative Examples

In this section, two examples are presented to show the application and effectiveness of the proposed method.

Example 1: It is from [1]. It is a 2-cluster tool composed of two single-arm cluster tools. The activity time is as follows: for C_1, $\alpha_{10} = 0$ (the loadlocks), $\alpha_{11} = 45$s, $\alpha_{12} = 0$ (the outgoing buffer), $\alpha_{13} = 5$s, $\alpha_{14} = 5$s, $\lambda_1 = 2$s, and $\mu_1 = 6$s; and for C_2, $\alpha_{20} = 0$ (the incoming buffer), $\alpha_{21} = 80$s, $\alpha_{22} = 80$s, $\alpha_{23} = 75$s, $\alpha_{24} = 77$s, $\lambda_2 = 3$s, and $\mu_2 = 4$s. The wafer processing route is LL $\rightarrow PS_{11} \rightarrow PS_{12}$ (PS_{20}) $\rightarrow PS_{21} \rightarrow PS_{22} \rightarrow PS_{23} \rightarrow PS_{24} \rightarrow PS_{20}$ (PS_{12}) $\rightarrow PS_{13} \rightarrow PS_{14} \rightarrow$ LL.

For C_1, we have $\eta_{10} = \alpha_{10} + 4\lambda_1 + 3\mu_1 = \eta_{12} = \alpha_{12} + 4\lambda_1 + 3\mu_1 = 0 + 4\times2 + 3\times6 = 26$s, $\eta_{11} = 71$s, $\eta_{13} = \eta_{14} = 31$s, and $\psi_{11} = 2(n[1] + 1)(\mu_1 + \lambda_1) = 2 \times (4 + 1)(2 + 6) = 80$s. Hence, $\Pi_1 = 80$ and C_1 is transport-bound. For C_2, we have $\eta_{20} = \alpha_{20} + 4\lambda_2 + 3\mu_2 = 0 + 4\times3 + 3\times4 = 24$s, $\eta_{21} = \eta_{22} = 104$, $\eta_{23} = 99$s, $\eta_{24} = 101$s, and $\psi_{21} = 2(n[2] + 1)(\mu_2 + \lambda_2) = 2 \times (4 + 1)(3 + 4) = 70$s. Hence, $\Pi_2 = 104$ and C_2 is process-bound. Because $\Pi_2 > \Pi_1$ and C_2 is process-bound, the multi-cluster is process-dominant. The lower bound of cycle time is $\Pi_2 = 104$.

According to the method proposed in this paper, let $\Theta = \Pi_2 = 104$. Then, we have $\psi_{12} = \Theta - \psi_{11} = 24$s and $\psi_{22} = \Theta - \psi_{21} = 34$s. This implies that we need to allocate 24s into ω_{1j}'s for R_1 in C_1, and 34s into ω_{2j}'s for R_2 in C_2, $j \in \{0, 1, 2, 3, 4\}$. Then, by using Algorithm 4.1, ω_{ij}'s are obtained as $\omega_{10} = \omega_{12} = \omega_{13} = \omega_{14} = 0$, $\omega_{11} = 24$, $\omega_{20} = \omega_{21} = 0$, $\omega_{22} = 5$, $\omega_{23} = 3$ and $\omega_{24} = 26$. In this way, a one-wafer periodic schedule is found. With $\Theta = \tau_{20} + 4\lambda_2 + 3\mu_2 + \omega_{24}$, $\tau_{20} = \Theta - (4\lambda_2 + 3\mu_2 + \omega_{24}) = 104 - (24 + 26) = 54 > (4\lambda_1 + 3\mu_1) = 26$, the condition given in Theorem 4.3 is satisfied, or the schedule obtained is optimal and its cycle time is 104s, the lower bound, which justifies the claim given in Section I. For this example, it should be noticed that an optimal 4-wafer cyclic schedule with cycle time 110.75s is obtained by the method given in [1]. The obtained cycle time outperforms the existing method [1] by 6.09%.

Example 2: It is from [2]. It is a 3-cluster tool composed of three single-arm cluster tools. The activity time is as follows: for C_1, $\alpha_{10} = 0$ (the loadlocks), $\alpha_{11} = 34$s, $\alpha_{12} = 0$ (the outgoing buffer), $\alpha_{13} = 31$s, $\alpha_{14} = 4$s, $\lambda_1 = 10$s, and $\mu_1 = 1$s; for C_2, $\alpha_{20} = 0$ (the incoming buffer), $\alpha_{21} = 82$s, $\alpha_{22} = 0$ (the outgoing buffer), $\alpha_{23} = 54$s, $\alpha_{24} = 12$s, $\lambda_2 = 7$s, and $\mu_2 = 1$s; and for C_3, $\alpha_{30} = 0$ (the incoming buffer), $\alpha_{31} = 54$s, $\alpha_{32} = 38$s, $\alpha_{33} = 91$s, $\alpha_{34} = 90$s, $\lambda_3 = 3$s, and $\mu_3 = 1$s. The

wafer processing route is LL→ PS_{11}→ PS_{12} (PS_{20}) → PS_{21} → PS_{22} → PS_{31} → PS_{32} → PS_{33} → PS_{34} → PS_{30} (PS_{22}) → PS_{23}→ PS_{24} → PS_{20} (PS_{12})→ PS_{13}→ PS_{14} → LL.

For C_1, we have $\eta_{10} = \alpha_{10} + 4\lambda_1 + 3\mu_1 = \eta_{12} = \alpha_{12} + 4\lambda_1 + 3\mu_1 = 0 + 4\times10 + 3\times1 = 43$s, $\eta_{11} = 77$s, $\eta_{13} = 74$s, $\eta_{14} = 47$s, and $\psi_{11} = 2(n[1] + 1)(\mu_1 + \lambda_1) = 2 \times (4 + 1) \times (10 + 1) = 110$s. Hence, $\Pi_1 = 110$ and C_1 is transport-bound. For C_2, we have $\eta_{20} = \alpha_{20} + 4\lambda_2 + 3\mu_2 = \eta_{22} = \alpha_{22} + 4\lambda_2 + 3\mu_2 = 0 + 4\times7 + 3\times1 = 31$s, $\eta_{21} = 113$s, $\eta_{23} = 85$s, $\eta_{24} = 43$s, and $\psi_{21} = 2(n[2] + 1)(\mu_2 + \lambda_2) = 2 \times (4 + 1) \times (7 + 1) = 80$s. Hence, $\Pi_2 = 113$ and C_2 is process-bound. For C_3, we have $\eta_{30} = \alpha_{30} + 4\lambda_3 + 3\mu_3 = 0 + 4\times3 + 3\times1 = 15$s, $\eta_{31} = 69$s, $\eta_{32} = 53$s, $\eta_{33} = 106$s, $\eta_{34} = 105$s, and $\psi_{31} = 2(n[3] + 1)(\mu_3 + \lambda_3) = 2 \times (4 + 1) \times (3 + 1) = 40$s. Hence, $\Pi_3 = 106$s and C_3 is also process-bound. Because $\Pi_2 > \Pi_1$, $\Pi_2 > \Pi_3$ and C_2 is process-bound, the multi-cluster is process-dominant.

According to the method proposed in this paper, let $\Theta = \Pi_2 = 113$. Then, we have $\psi_{12} = \Theta - \psi_{11} = 3$s, $\psi_{22} = \Theta - \psi_{21} = 33$s, and $\psi_{32} = \Theta - \psi_{31} = 73$s. This implies that we need to allocate 3s into ω_{1j}'s for R_1 in C_1, 33s into ω_{2j}'s for R_2 in C_2, and 73s into ω_{3j}'s for R_3 in C_3, $j \in \{0, 1, 2, 3, 4\}$. Then, by using Algorithm 4.1, we can find $\omega_{10} = 0$, $\omega_{11} = 3$, $\omega_{12} = \omega_{13} = \omega_{14} = 0$, $\omega_{20} = 0$, $\omega_{21} = 33$, $\omega_{22} = \omega_{23} = \omega_{24} = 0$, $\omega_{30} = 44$, $\omega_{31} = 29$, and $\omega_{32} = \omega_{33} = \omega_{34} = 0$. In this way, a one-wafer periodic schedule is found. By (3.6), $\tau_{30} = \Theta - (4\lambda_3 + 3\mu_3 + \omega_{34}) = 113 - 15 = 98 > 4\lambda_2 + 3\mu_2 = 31$, and $\tau_{20} = \Theta - (4\lambda_2 + 3\mu_2 + \omega_{24}) = 113 - 31 = 82 > 4\lambda_1 + 3\mu_1 = 43$. Hence, the condition given in Theorem 4.3 is satisfied, or the schedule obtained is optimal and its cycle time is 113s, which is the lower bound. For this example, in [2], an optimal 5-wafer periodic schedule is obtained with cycle time 114.8s that is greater than the lower bound.

6. CONCLUSIONS

With multiple robots, it is very challenging to schedule a multi-cluster tool to maximize the throughput. Up to now, there is no efficient method to find a schedule with the lower bound of cycle time for a multi-cluster tool. The industrial practitioners prefer one-wafer schedule because of its simplicity and easy implementation. This paper conducts a study on finding a one-wafer periodic schedule to obtain the lower-bound of cycle time for multi-cluster tools. It is found that, in scheduling a multi-cluster tool, the key question is how to determine the robots' waiting time. With this in mind, a resource-oriented Petri net model is developed for such a system, such that the robots' waiting time is well modeled. Furthermore, this model describes not only the behavior of its steady state but also the behavior of its initial and final transient processes. Based on the model, optimality conditions are found and the scheduling problem is reduced to the determination of the robots' waiting time. By the derived conditions, its optimal one-wafer periodic schedule for a multi-cluster tool can be very efficiently obtained by scheduling its individual cluster tools one by one. Furthermore, with the same model, an effective method is proposed to implement the obtained optimal schedule. Such results are never obtained to the best knowledge of the authors.

It should be pointed out that a schedule with the lower bound of cycle time can be found only if the conditions given in this paper are satisfied. Although the derived conditions can be satisfiable for many industrial applications, there may be cases such that they are not. Thus, it is our future work to search for an efficient scheduling method to obtain optimal one-wafer cyclic schedule for such cases. For some wafer fabrication processes, there are wafer residency time constraints. In this paper, we do not consider them. Thus, it is also our future work to schedule such multi-cluster tools with wafer residency time constraints.

REFERENCES

[1] W. K. Chan, J. G. Yi, and S. W. Ding, "Optimal Scheduling of Multicluster Tools with Constant Robot Moving Times, Part I: Two-Cluster Analysis," *IEEE Transactions on Automation Science and Engineering,* vol. 8, no.1, pp. 5-16, Jan. 2011.

[2] W. K. Chan, J. G. Yi, S. W. Ding, and D. Z. Song, "Optimal Scheduling of Multicluster Tools with Constant Robot Moving Times, Part II: Tree-Like Topology Configurations," *IEEE Transactions on Automation Science and Engineering,* vol. 8, no.1, pp. 17-28, Jan. 2011.

[3] M. Dawande, C. Sriskandarajah, and S. P. Sethi, "On Throughput Maximization in Constant Travel-time Robotic Cells," *Manufacture Service Operation Manage,* vol. 4, no. 4, pp. 296–312, 2002.

[4] M. Dawande, "Throughput optimization in robotic cells." Vol. 101. Springer, 2007

[5] S. W. Ding, J. G. Yi, and M. T. Zhang, "Multicluster Tools Scheduling: an Integrated Event Graph and Network Model Approach," *IEEE Transactions on Semiconductor Manufacturing,* vol. 19, no.3, pp. 339 - 351, Aug. 2006.

[6] I. Drobouchevitch, S. P. Sethi, and C. Sriskandarajah, "Scheduling Dual Gripper Robotic Cells: One-unit Cycles," *European Journal of Operational Research*, vol. 171, no. 2, pp. 598–631, Jun. 2006.

[7] D. Jevtic, "Method and Apparatus for Managing Scheduling a Multiple Cluster Tool," in *European Patent*. vol. 1,132,792(A2), 2001.

[8] D. Jevtic and S. Venkatesh, "Method and Apparatus for Scheduling Wafer Processing within a Multiple Chamber Semiconductor Wafer Processing Tool Having a Multiple Blade Robot," in *U.S. Patent*. vol. 6,224,638, 2001.

[9] J.-H. Kim, T.-E. Lee, H.-Y. Lee, and D.-B. Park, "Scheduling analysis of timed-constrained dual-armed cluster tools," *IEEE Transactions on Semiconductor Manufacturing*, vol. 16, no. 3, pp. 521-534, Aug. 2003.

[10] T.-E. Lee, H.-Y. Lee, and Y.-H. Shin, "Workload balancing and scheduling of a single-armed cluster tool," in *Proceedings of the 5th APIEMS Conference*, Gold Coast, Australia, pp. 1–15, 2004.

[11] T.-E. Lee and S.-H. Park, "An extended event graph with negative places and tokens for timed window constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 4, pp. 319-332, Oct. 2005.

[12] M.-J. Lopez and S.-C. Wood, "Systems of multiple cluster tools - configuration, reliability, and performance," *IEEE Transactions on Semiconductor Manufacturing*, vol. 16, no. 2, pp. 170-178, May 2003.

[13] T. L. Perkinson, P. K. MacLarty, R. S. Gyurcsik, and R. K. Cavin, III, "Single-wafer cluster tool performance: An analysis of throughput," *IEEE Transactions on Semiconductor Manufacturing*, vol. 7, no.3, 369-373, Aug. 1994.

[14] T. L. Perkinson, R. S. Gyurcsik, and P. K. MacLarty, "Single-wafer cluster tool performance: An analysis of the effects of redundant chambers and revisitations sequences on throughput," *IEEE Transactions on Semiconductor Manufacturing*, vol. 9, pp. 384-400, Aug. 1996.

[15] Y. Qiao, N. Q. Wu, and M. C. Zhou, Real-time scheduling of single-arm cluster tools subject to residency time constraints and bounded activity time variation, *IEEE Transactions on Automation Science and Engineering*, vo. 9, no. 3, pp. 564-577, Jul. 2012.

[16] Y. Qiao, N. Q. Wu, and M. C. Zhou, Petri net modeling and wafer sojourn time analysis of single-arm cluster tools with residency time constraints and activity time variation, *IEEE Transactions on Semiconductor manufacturing*, vol. 25. no. 3, pp. 432-446, Aug. 2012.

[17] Y. Qiao, N. Q. Wu, and M. C. Zhou, "A Petri net-based novel scheduling approach and its cycle time analysis for dual-arm cluster tools with wafer revisiting," to appear in *IEEE Transactions on Semiconductor manufacturing*.

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

[19] N. Q. Wu, "Necessary and Sufficient Conditions for Deadlock-free Operation in Flexible Manufacturing Systems Using a Colored Petri Net Model," *IEEE Transaction on Systems, Man, and Cybernetics, Part C,* vol. 29, no. 2, pp. 192-204, 1999.

[20] N. Q. Wu, C. B. Chu, F. Chu, and M. C. Zhou, "A Petri net method for schedulability and scheduling problems in single-arm cluster tools with wafer residency time constraints," *IEEE Transactions on Semiconductor Manufacturing,* vol. 21, no. 2, pp. 224 - 237, May 2008.

[21] N. Q. Wu, F. Chu, C. Chu, and M. Zhou, "Petri Net-Based Scheduling of Single-Arm Cluster Tools With Reentrant Atomic Layer Deposition Processes," *IEEE Transactions on Automation Science and Engineering,* vol. 8, no. 1, pp. 42-55, Jan. 2011.

[22] N. Q. Wu, F. Chu, C. B. Chu, and M. C. Zhou, "Petri net modeling and cycle time analysis of dual-arm cluster tools with wafer revisiting,", to appear in *IEEE Transactions on Systems*, *Man*, *& Cybernetics*: *Systems*.

[23] N. Q. Wu and M. C. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems based on a Petri net model," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 658-669, Oct. 2001.

[24] N. Q. Wu and M. C. Zhou, "Modeling and deadlock control of automated guided vehicle systems," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 1, pp. 50-57, 2004.

[25] N. Q. Wu and M. C. Zhou, *System modeling and control with resource-oriented Petri nets*, CRC Press, Taylor & Francis Group, New York, October 2009.

[26] N. Q. Wu and M. C. Zhou, "Process vs resource-oriented Petri net modeling of automated manufacturing systems," *Asian Journal of Control*, vol. 12, no. 3, pp. 267-280, 2010.

[27] N. Q. Wu and M. Zhou, "Analysis of wafer sojourn time in dual-arm cluster tools with residency time constraint and activity time variation," *IEEE Transactions on Semiconductor Manufacturing,* vol. 23, no. 1, pp. 53-64, Feb. 2010.

[28] N. Q. Wu and M. C. Zhou, "A closed-form solution for schedulability and optimal scheduling of dual-arm cluster tools based on steady schedule analysis," *IEEE Transactions on Automation Science and Engineering,* vol. 7, no. 2, pp. 303-315, Apr. 2010.

[29] N. Q. Wu and M. C. Zhou, "Modeling, analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on Petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 446-454, Apr. 2012.

[30] N. Q. Wu and M. C. Zhou, "Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 1, pp. 203-209, Jan. 2012.

[31] N. Q. Wu, M. C. Zhou, F. Chu, and C. B. Chu, "A Petri-net-based scheduling strategy for dual-arm cluster tools with wafer revisiting," to appear in *IEEE Transactions on Systems*, *Man*, *& Cybernetics*: *Systems*.

[32] N. Q. Wu, M. C. Zhou, and G. Hu, "One-step look-ahead maximally permissive deadlock control of AMS by using Petri net," to appear in *ACM Transactions on Embedded Computing Systems*.

[33] J. Yi, S. Ding, and D. Song, "Steady-state throughput and scheduling analysis of multi-cluster tools for semiconductor manufacturing: A decomposition approach," in *Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pp. 292–298, 2005.

[34] J. G. Yi, S. W. Ding, D. Z. Song, and M. T. Zhang, "Steady-State Throughput and Scheduling Analysis of Multi-Cluster Tools for Semiconductor Manufacturing: A Decomposition Approach," *IEEE Transactions on Automation Science and Engineering,* vol. 5, no. 2, pp. 321-336, Apr. 2008.

[35] M. C. Zhou and M. D. Jeng, "Modeling, Analysis, Simulation, Scheduling, and Control of Semiconductor Manufacturing Systems: A Petri Net Approach," *IEEE Transactions on Semiconductor Manufacturing,* vol. 11, no. 3, pp. 333-357, Aug. 1998.

[36] W. M. Zuberek, "Timed Petri nets in modeling and analysis of cluster tools," *IEEE Transactions on Robotics Automation*, vol. 17, no. 5, pp. 562-575, Oct. 2001.

QingHua Zhu received the B.S. degree in computer science from Jiangxi Normal University, Nanchang, China, in 1997, and the M.S. degree in transportation information engineering and control from East China Jiao Tong University, Nanchang, China, in 2003.

He had been with Jiangxi University of Finance and Economics, China, for three years. In 2003, he joined Guangdong University of Technology, where he is an Assistant Professor with the Department of Computer Engineering, School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China. His current research interests include discrete event systems, production planning and scheduling, and Petri nets.

Yan Qiao received the B. S. degree in Industrial Engineering from Guangdong University of Technology, Guangzhou, China, in 2009. Since 2009, he has been a doctoral graduate student at the Department of Industrial Engineering, School of Electro-Mechanical Engineering, Guangdong University of Technology. He was the 2011 IEEE CASE QSI Best Application Paper Award Finalist and received the Best Student Paper Award in 2012 IEEE ICNSC. His research interests include discrete event systems, production planning, Petri nets, scheduling and control.