Optimal Scheduling of k-Unit Production of Cluster Tools with Single-Blade Robots

Wai-Kin (Victor) Chan, Jingang Yi, Shengwei Ding, and Dezhen Song

***Abstract*— The main challenge in scheduling multi-cluster tools in semiconductor manufacturing is the interactions among clusters. These interactions create a k-unit optimal production that do not exist in single-cluster tools. This paper analyzes optimal scheduling of k-unit cycle production of multi-cluster tools with single-blade robots. A resource-based method is used to analytically derive closed-form expressions for the minimal cycle time of a multi-cluster tool. Conditions for decoupling multi-cluster tools and optimality conditions for the widely used pull schedule are also presented. An example from industry production is used to illustrate the derived formula and decoupling conditions.**

I. Introduction

With the increasing complexity of semiconductor manufacturing processes, multi-cluster tools are used to accommodate the industry needs [1]. For a multi-cluster tool, such as the 2-cluster tools shown in Fig. 1(a), there are multiple robots to transfer wafers among various clusters, and each robot serves within one cluster. Each cluster consists of process modules (PMs), transport modules (robots), and load locks (cassette modules). Each wafer is picked up by a transfer robot and moved to PMs according to predefined routing sequences (recipes). The robot in a cluster tool can be single-blade or double-blade. A single-blade robot can hold only one wafer at a time.

The goal of this paper is to provide a methodology to analyze the k-unit production cycle time and to optimally schedule robots in multi-cluster tools. We consider an M-multi-cluster tool that is composed of M single clusters connected in a tree-like layout, that is, no loop inter-connection. Only one- or two-wafer capacity inter-cluster buffers are considered here. Fig. 1(b) shows an example of a 10-cluster tool.

A multi-cluster tool can be considered as several interconnected single clusters, or robotic cells. The main complexity of scheduling of a multi-cluster tool comes from interaction dependencies among individual clusters. The "pull" (or reverse) schedule for single-blade robots is discussed as the optimal schedule for single cluster tools [1]. In the pull schedule, the robot is sequentially moving wafers

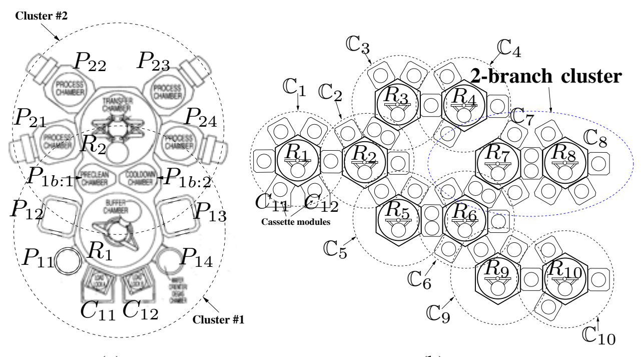

Fig. 1. A schematic of (a) a two-cluster tool and (b) an inter-connected 10-cluster tool

from one PM to the next PM. In [1], a polynomial time algorithm is provided for finding the optimal one-unit cycle schedule under constant robot moving time. Heuristics and approximation algorithms dealing with k-unit cycles in a single robotic cell have been studied in [2], [3].

For multi-cluster tools, a few results have been reported recently. In [4], a robotic cell with three single-gripper robots for semiconductor manufacturing is presented, and the authors compared the throughput of the pull strategy by simulation. In [5], an integrated event graph and network model is used to find all optimal schedules for a multi-cluster tool. The recent study [6] presents an analytical scheduling scheme for multi-cluster tools using a decomposition approach. Chan *et al.* [7] extend the results in [6] with a consideration of non-zero robot moving time. However, neither [7] nor [6] does consider the multiple-unit cycles.

The focus of this paper is on the throughput and robot scheduling of k-unit ($k \geq 1$) production of multi-cluster tools. This paper is an extension of the previous work in [5]–[7]. The contributions of this paper are twofold. First, we use a concept of resource cycles that is first proposed in [7] to quantify the dependencies among clusters in a multi-cluster tool. The unified treatment of all resources allows us to identify the interactions among single clusters. Such interactions can cause that every single cluster repeats its one-unit schedule for k times before the whole multi-cluster tool returns to the original state at the beginning of these one-unit schedules. We call this repeated schedule as k-unit cycle production. To our knowledge, these phenomena have not been studied in the literature. Second, we establish conditions under which a multi-cluster tool can be decoupled into single clusters so that existing efficient algorithms for single cluster tools can be used to find optimal schedules.

The rest of the paper is organized as follows. In Section II,

W.-K. Chan is with the Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA. E-mail: chanw@rpi.edu.

J. Yi is with the Department of Mechanical Engineering, San Diego State University, San Diego, CA 92182 USA. E-mail: jgyi@mail.sdsu.edu.

S. Ding is with the Department of Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley, CA 94720 USA. E-mail: dingsw@cal.berkeley.edu.

D. Song is with the Department of Computer Science, Texas A&M University, College Station, TX 77843, USA. E-mail: dzsong@cs.tamu.edu.

[1]For example, multi-cluster tools can be found at http://www.amat.com/products

we introduce notations and the concept of resource cycle for single-cluster tools. In Section II, we present the optimal scheduling analysis. The decoupling conditions are discussed in Section IV. An industrial example of a multi-cluster tool is presented in Section V before we conclude the paper.

II. Resource-Based Cluster-Tool Scheduling

A. Notations and problem statement

We consider the following assumptions for M-multi-cluster tools: (1) All wafers follow an identical flow that visits each PM only once (multiple parallel PMs is considered as one PM); (2) The cassette modules always have wafers/spaces for robot to pick or place at any time; (3) All the activity times are deterministic; and (4) Buffer modules have either one- or two-wafer capacity. We define an m-branch-cluster tool (Fig. 2) as a branch of an M-multi-cluster tool. An m-branch-cluster tool consists of m single cluster tools in series, each of which connects to exactly two neighboring single clusters, except the leaf (ending) and the head (starting) single clusters, which connect to only one neighboring single cluster. We also use the word "job" to represent "wafer" in all subsequent discussion.

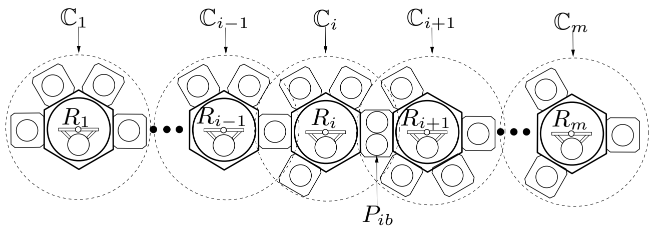

Fig. 2. A schematic of an inter-connected m-branch cluster.

We assume that single cluster $\mathbb{C}_i$ [2], $i = 1, \ldots, m$, consists of c_i number of PMs, denoted by P_{ij}, $j = 1, \ldots, c_i$, and a single-blade robot R_i. Here we consider the buffer module connecting $\mathbb{C}_i$ and $\mathbb{C}_{i+1}$ as one of the PMs belonging to $\mathbb{C}_i$, and we denote b_i as the index of the buffer module P_{ib_i}. For a single-space buffer module, both the inlet and outlet jobs share the same space. For a double-space buffer module, $P_{ib_i:1}$ and $P_{ib_i:2}$, respectively, the first (from $\mathbb{C}_i$ and $\mathbb{C}_{i+1}$) and second buffer (from $\mathbb{C}_{i+1}$ back to $\mathbb{C}_i$) spaces of the buffer module. For $\mathbb{C}_i$, the processing time for buffer P_{ib_i} is denoted by t_{ib_i}. For $\mathbb{C}_{i+1}$, the buffer module serves as cassette modules. The index of this cassette module is "0" and the processing time is denoted by $t_{i+1,0}$. No real wafer processing is taken in all buffer modules, namely, $t_{ib_i} = t_{i+1,0} = 0$. We however keep these notations for capturing the dependencies among clusters in later sections.

For cluster $\mathbb{C}_i$, let t_{ij} be the processing time for module P_{ij}, ϵ_i be the time to load/unload a job to/from a PM, and δ_i be the robot move time between any pair of modules/buffer/cassette within $\mathbb{C}_i$. To facilitate later discussion, we define $\alpha_{ij} = 2\epsilon_i + \delta_i + t_{ij}$, $\beta_i = 2(\epsilon_i + \delta_i)$, $\beta_i^{\delta_i} = \beta_i - \delta_i$, $\alpha_{ij}^0 = \alpha_{ij} - t_{ij}$. We assume that the wafer processing at each PM starts right after the robot places an unprocessed wafer inside that PM. As a result, we only consider the sequencing and scheduling of robot movement activities. We define an activity A_{ij} associated with P_{ij} in $\mathbb{C}_i$ as three consecutive actions taken by robot R_i: (1) unloading a job from P_{ij}, (2) traveling from P_{ij} to $P_{i,j+1}$, and (3) loading this job into $P_{i,j+1}$. Activities A_{i0} and A_{ic_i} represents the actions of loading/unloading a job from or to cassette modules. We take the same treatment for buffer modules with activities A_{i,b_i-1} and A_{ib_i}.

We consider a *one-unit wafer cycle* as a sequence of feasible activities consisting of every activity exactly once and during which, exactly one job is imported from and exported to the cluster tool. The *one-unit cycle time* (or simply cycle time) is the minimum time required to complete the one-unit wafer cycle. For a single-cluster tool, after a one-unit cycle, the tool returns to the same state as the beginning of the cycle. However, this is not always true for multi-cluster tools. A concept of resource cycle will be used to capture such repeated cycles.

A one-unit cycle schedule in $\mathbb{C}_i$ is determined by the robot movement sequence $\pi_i = (A_{ij_0} A_{ij_1} \cdots A_{ij_{c_i}})$. An optimal schedule π_i^* maximizes the throughput and therefore minimizes the cycle time $T_i(\pi_i)$. We can view a one-unit cycle robot schedule for an M-multi-cluster tool is determined by two steps: the first step considers only a combination of all individual robot schedule $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_M)$. The second step is to coordinate all π_i. To capture the robot coordination among clusters, we consider the distribution of jobs (wafers) among clusters. We denote the job distribution among an M-multi-cluster tool as $\mathbf{n} = (n_1, n_2, \ldots, n_M)$, where n_i is the number of jobs in $\mathbb{C}_i$. We denote $T_{i,\ldots,m}(\pi_i, \ldots, \pi_m; n_i, \ldots, n_m)$ the cycle time of an $(m-i+1)$-branch cluster tool with the first to the ending cluster tools indexed from i to m and under schedule $(\pi_i, \ldots, \pi_m)$ and job distribution $(n_i, \ldots, n_m)$.

B. Resource-based optimal schedules for single-cluster tool

For $\mathbb{C}_i$, we define the *push* strategy as a schedule with activity indexes $(A_{i0} A_{i1} \cdots A_{ic_i})$, namely, in an increasing order starting with $A_{i0} A_{i1}$, and the *pull* strategy as a schedule with activity indexes $(A_{i0} A_{ic_i} \cdots A_{i1})$ in an decreasing order starting with $A_{i0} A_{ic_i}$. We denote the cycle time for the push schedule as $T_i^F = \beta_i + \sum_{l \in \{1, \ldots, c_i\}} \alpha_{il}$. In [7], we define an *activity chain* in a single cluster is a sequence of activities (excluding A_{i0}) with consecutively increasing order indexes appear (not necessary adjacently) in a schedule. Activity A_{i0} itself is also an activity chain. We have the following result [3].

Lemma 1: Given a schedule $\pi_i = \left(A_{ij_0} A_{ij_1} \cdots A_{ij_{c_i}}\right)$ for a single-cluster tool, the number of activity chains is equal to the number of jobs n_i in the tool under π_i.

For example, for schedule $\pi_i = (A_{i0} A_{i3} A_{i1} A_{i2})$, $n_i = 2$ since, excluding A_{i0}, there are only two activity chains in

[2] As [6], we define the decoupled single-cluster $\mathbb{C}_i$ as the ith single cluster detached from a multi-cluster tool by replacing the buffer modules with virtual PMs and infinite number of jobs and spaces at the input/output module. With a slight abuse of notation, we use $\mathbb{C}_i$ to indicate the ith cluster of a multi-cluster tool and the decoupled ith cluster without confusion.

[3] Due to the page limit, all proofs of lemmas and theorems are omitted in this paper. The details of all proofs can be found in [8].

π_i. We note that n_i is a function of π_i. Also, $n_i = 1$ under the push schedule and $n_i = c_i$ under the pull schedule.

We also define a *resource* as a physical object performing certain timed activities, for example, PMs, robots, buffers, and jobs. The *one-unit resource cycle* (or simply resource cycle) of a resource is a sequence of activities that, (1) starting from a particular state, returns the resource to its original state if the resource is a PM, robot, or buffer, or (2) starting from a particular activity chain, advances the resource to the next activity chain if the resource is a job. The *resource cycle time* (RCT) of resource Q, denoted by $RCT(Q)$, is the minimum cycle time duration for a resource to perform all the activities in its own resource cycle without extra waiting for other resource cycles.

In a one-unit cycle, since A_{i0} advances a new job to A_{i1} and each of the other n_i jobs advances to the next activity chain, all the activities chains repeat exactly once. Given a schedule $\pi_i = \left(A_{ij_0} A_{ij_1} \cdots A_{ij_{c_i}}\right)$, the resource cycle for P_{ij} is the sequence of activities from A_{ij} to $A_{i,j-1}$ inclusively, namely, $RC(P_{ij}) = (A_{ij} A_{ij_1} \cdots A_{ij_{k(\pi_i)}} A_{i,j-1})$. Let $IA(P_{ij}) = \{j, j_1, \ldots, j_{k(\pi_i)}, j-1\}$ be the index set of the activities in the resource cycle of P_{ij}.

From [7], we know that all process and buffer modules can be categorized into two types of resources: $\Lambda_i^P = \{j | (A_{i,j-1} A_{ij}) \not\subseteq \pi_i\}$, and $\Lambda_i^R = \{j | (A_{i,j-1} A_{ij}) \subseteq \pi_i\}$, where Λ_i^P is the index set of resources whose resource cycle is different from the robot cycle, and Λ_i^R is the index set of resources whose resource cycle times coincide with that of the robot. We define the *unavoidable activity* set $UA(P_{ij})$, as the index set of js and the activities contained in both $IA(P_{ij})$ and Λ^R, namely, $UA(P_{ij}) = \{IA(P_{ij}) \cap \Lambda_i^R\} \cup \{j\}$. From [7], we have

$$RCT(P_{ij}) = (|IA(P_{ij})| - |UA(P_{ij})|)\beta_i + \sum_{k \in UA(P_{ij})} \alpha_{ik},$$

where $|\cdot|$ denotes the cardinality of a set, and

$$RCT(R_i) = (c_i + 1 - |\Lambda_i^R|)\beta_i + \sum_{j \in \Lambda_i^R} \alpha_{ij}.$$

The following lemma from [7] gives a lower bound of the tool's cycle time.

Lemma 2: For a single-cluster tool, the one-unit cycle time $T_i(\pi_i)$ under schedule π_i satisfies

$$T_i(\pi_i) \geq \max\left\{\max_{j \in \Lambda_i^P}\{RCT(P_{ij})\}, RCT(R_i)\}\right\}. \quad (1)$$

So far our discussion on the resource cycle of a job only focuses on a one-unit cycle. We now describe the complete resource cycle of a job. We define the *complete-job-resource cycle time* as the minimal total time for a job to be processed and exit the whole single-cluster tool without extra waiting inside the tool. The complete-job-resource cycle time is equal to the sum of the times for all activities chains, which is also equal to the cycle time T_i^F of the push schedule. Since the total number of jobs produced during a complete-job-resource cycle time is n_i, we also call the complete-job-resource cycle time as *n-unit job-resource cycle time* and the corresponding cycle is called the *n-unit job-resource cycle*. The *average n-unit job-resource cycle time* (or simply *average job-resource cycle time*) (AJRCT) is the average time for a job to go through one activity chain, namely, $\frac{T_i^F}{n_i}$.

For a single-cluster tool, the AJRCT cannot dominate the whole cycle time. Intuitively, this observation results from the fact that the AJRCT in every one-unit cycle is less than the one-unit cycle time due to no congestion (blocking) is assumed at the input/output module. Because the AJRCT never dominates the whole one-unit cycle time, most literature do not include them in computing the cycle time for single-cluster tools; see [1]. However, the AJRCT would be useful in in the analysis of multi-cluster tools dealing with multiple parts.

In [1], a concept of basic cycles is proposed. A basic cycle dominates all other schedules in the sense of having smaller cycle times and is a schedule with activity indexes in decreasing order except those in Λ_i^R. We have the following results.

Theorem 1: For a single cluster under a *basic cycle* schedule π_i, $|IA(P_{ij})| = |UA(P_{ij})| + 1$, for all $j \in \Lambda_i^P$. Moreover, the one-unit cycle time $T_i(\pi_i)$ is $T_i(\pi_i) = \max\left\{\max_{j \in \Lambda_i^P}\{RCT(P_{ij})\}, RCT(R_i)\}\right\}$.

III. Scheduling of Multi-Cluster Tools

In this section, we first discuss the job distribution among a multi-cluster tool. Then we discuss the minimum cycle time analysis for an m-branch and M-cluster tool.

A. Job distribution in a multi-cluster tool

In an m-cluster tool, we need to specify the job distribution to completely determine the cycle time. We represent the number of jobs in cluster $\mathbb{C}_i$ in terms of n_i, the number of jobs in the corresponding decoupled single-cluster tool. Moreover, the jobs in the buffer module P_{ib_i} is considered as jobs belonging to $\mathbb{C}_i$, instead of $\mathbb{C}_{i+1}$. It is necessary to explain when and how the job distribution in $\mathbb{C}_i$ is defined. First, the number of jobs in $\mathbb{C}_i$ in a multi-cluster tool is equal to the number of jobs in that single cluster during the steady state operation after A_{i0} and before A_{ic_i}. Second, it would be helpful to consider the initial job location. For $\mathbb{C}_i$ with a double-space buffer, let n_i be the number of jobs that is determined by Lemma 1. The initial job location is to have one job only in the beginning PM of each activity chain except the chain that includes the buffer: (1) When the job distribution is $n_i - 1$, then there is no job in the activity chain that contains the buffer; (2) When the job distribution is n_i, then the activity chain that contains the buffer should have one job in the first buffer space only; and (3) When the job distribution is $n_i + 1$, then the activity chain that contains the buffer should have totally two jobs in the two buffer spaces. For $\mathbb{C}_i$ with a single-space buffer, the job distribution and initial location is the same as in Case (1) above.

Let us consider a 2-branch cluster $(\mathbb{C}_1, \mathbb{C}_2)$ under schedule (π_1, π_2). Let n_i be the number of jobs in $\mathbb{C}_i$ (when it is decoupled from the other one) under schedule π_i, $i = 1, 2$ respectively. The number of jobs could vary under (π_1, π_2).

If P_{1b_1} is single-space, then the total number of jobs $N = n_1 + n_2 - 1$ is distributed in $\mathbb{C}_1$ and $\mathbb{C}_2$ as $(n_1 - 1, n_2)$; if P_{1b_1} is double-space, N can then be the following three cases: (a) $N = n_1 + n_2 - 1$ and distributed as $(n_1 - 1, n_2)$; (b) $N = n_1 + n_2$ and distributed as (n_1, n_2); and (c) $N = n_1 + n_2 + 1$ and distributed as $(n_1 + 1, n_2)$. The analysis is given in [8].

B. Minimum cycle time and optimal schedules

We define the *minimal cycle time* is the smallest cycle time over all possible job distributions under the same given schedule in a multi-cluster tool. Let $T_i^0(\pi_i; n_i)$ and $T_i^{t_{S_i}}(\pi_i; n_i)$ denote the cycle times of the decoupled cluster $\mathbb{C}_i$ with zero processing time at P_{ib_i} (i.e., $t_{ib_i} = 0$) and processing time t_{S_i} at P_{ib_i} (i.e., $t_{ib_i} = t_{S_i}$), respectively. From [8], we show that the job distributions $n_1 - 1$ and $n_1 + 1$ in $\mathbb{C}_1$ can be treated as n_1, and $T_1^0(\pi_1; n_1 - 1)$ and $T_1^0(\pi_1; n_1 + 1)$ are thus equal to $T_1^0(\pi_1; n_1)$. In a multi-cluster tool, although all the robots can work in a coordinated fashion to improve the over-all cycle time, such a collaborative robot movement cannot reduce any single-cluster's cycle time more than what the decoupled cluster with $t_{ib_i} = 0$ can reach. This observation is summarized in the next lemma.

Lemma 3: Let $T_i(\pi_i)$ be the cycle time for $\mathbb{C}_i$ when operates collaboratively with all other clusters in the multi-cluster tool, then $T_i(\pi_i) \geq T_i^0(\pi_i)$ under any π_i.

If we attempt to schedule the multi-cluster tool using the decoupled single-cluster tools with all buffer modules replaced by zero-processing time modules, it is unlikely to obtain a cycle time equal to the lower-bound because some clusters are forced to synchronize with the adjacent clusters. Therefore, it is necessary to take into account the interactions among all clusters.

Consider a 2-cluster tool consisting of $\mathbb{C}_1$ and $\mathbb{C}_2$ with $\mathbb{C}_2$ being the leaf cluster tool. We drop n_2 from $T_2^0(\pi_2)$ because n_2 is completely determined by π_2 as shown in Lemma 1. Let k_L be the largest PM index (if exists) in Λ_1^P smaller than k in π_1 and k_R be the smallest PM index (if exists) in Λ_1^P greater than k in π_1. From $\mathbb{C}_2$'s point of view, P_{1b_1} is its input/output module P_{20}. After R_2 unloads a job from it, it will take a certain time delay before R_2 loads a job to it. This delay makes P_{20}, from $\mathbb{C}_1$'s point of view, a resource with a recourse cycle time as $RCT(P_{20})$. Also, let $RCT^0(P_{20}) = RCT(P_{20}) - t_{20}$ and the following theorem gives a formula for the cycle time of a 2-cluster tool with a single-space buffer.

Theorem 2: For a 2-cluster tool $\mathbb{C}_1$ and $\mathbb{C}_2$ connected by a single-space buffer P_{1b_1} and under schedule $\boldsymbol{\pi} = (\pi_1, \pi_2)$, the minimal cycle time is

$$\begin{aligned} &T_{1,2}(\pi_1, \pi_2; n_1 - 1, n_2) = \\ &\max\{T_1^{t_{S_1}}(\pi_1; n_1 - 1), T_2^0(\pi_2), K_{12}\}, \end{aligned}$$

where $T_1^{t_{S_1}}(\pi_1; n_1 - 1) = T_1(\pi_1; n_1)|_{t_{1b_1} = t_{S_1}}$, $t_{S_1} = RCT^0(P_{20}) = \beta_2^{\delta_2} + \sum_{l=1}^{p} \alpha_{2l} + \beta_2 + \sum_{l=q+1}^{c_2} \alpha_{2l}$, $p = \max\{p : A_{20}A_{21} \cdots A_{2p} \subseteq \pi_2\}$, $q = \min\{q : A_{2q}A_{2,q+1} \cdots A_{2,c_2-1}A_{2,c_2} \subseteq \pi_2\}$,

$$K_{12} = \begin{cases} \frac{RCT^0(P_{1b_1}) + T_2^{F^{\delta_2}}}{n_2}, & \text{if } b_1 \in \Lambda_1^P, \\ \max\left\{ \frac{RCT^0(Q) + T_2^{F^{\delta_2}}}{n_2} : Q \in \{P_{1k_L}, P_{1k_R}, R_1\} \right\}, & \\ & \text{if } b_1 \in \Lambda_1^R, \end{cases}$$

$RCT^0(P_{1b_1}) = RCT(P_{1b_1})|_{t_{1b_1}=0}$, $RCT^0(Q) = RCT(Q)|_{t_{1b_1}=0}$, and $T_2^{F^{\delta_2}}(\pi_2) = \beta_2^{\delta_2} + \sum_{j=1}^{c_2} \alpha_{2j}$; if $p = 0$ ($q = c_2$), then the first (second) sum in t_{S_1} is zero.

Theorem 2 captures the dependencies between $\mathbb{C}_1$ and $\mathbb{C}_2$ analytically through two terms: $RCT^0(P_{1b_1}) + t_{S_1}$ and K_{12}. t_{S_1} can be viewed as P_{1b_1}'s "virtual processing time," which is equal to the time duration for R_2 to return a job to P_{1b_1} after R_2 unloads a job from it, that is, $RCT^0(P_{20})$. K_{12} is the AJRCT, representing the n-unit job-resource cycle, which is the phenomenon appearing in scheduling multi-cluster tools but not in single-cluster tools. The K_{12} term is due to interaction between the two single clusters. We can extend the results in Theorem 2 to an m-branch tool [8].

Corollary 1: For an m-branch-cluster tool under a schedule $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ and all buffers are single-space, the minimal cycle time is

$$\begin{aligned} &T_{1,\ldots,m}(\pi_1, \ldots, \pi_m) \\ &= \max\Big\{ \max_{1 \leq i \leq m-1} T_i^{t_{S_i}}(\pi_i), T_m^0(\pi_m), \max_{\substack{1 \leq i \leq m-1 \\ j > i}} K_{ij} \Big\}, \quad (2) \end{aligned}$$

where $T_i^{t_{S_i}}(\pi_i) = T_i(\pi_i)|_{t_{ib_i} = t_{S_i}}$, $t_{S_i} = RCT^0(P_{i+1,0})$, for $b_i \in \Lambda_i^P$, $i \neq m$, $j = i + 1, \ldots, m - 1$,

$$\begin{cases} K_{ij} = \frac{RCT^0(P_{ib_i}) + \sum_{k=i+1}^{j} T_k^{F^{0\delta_k}} + RCT^0(P_{k0})}{\sum_{k=i+1}^{j-1}(n_k - 1) + n_j}, \\ K_{im} = \frac{RCT^0(P_{ib_i}) + \sum_{k=i+1}^{m-1} T_k^{F^{0\delta_k}} + T_m^{F^{\delta_m}}}{\sum_{k=i+1}^{m-1}(n_k - 1) + n_m}, \end{cases}$$

$RCT^0(P_{ib_i}) = RCT(P_{ib_i})|_{t_{ib_i}=0}$, $RCT^0(P_{i0}) = \beta_i^{\delta_i} + \sum_{l=1}^{p_i} \alpha_{il} + \beta_i + \sum_{l=q_i+1}^{c_i} \alpha_{il}$, $p_i = \max\{p_i : A_{i0}A_{i1} \cdots A_{ip_i} \subseteq \pi_i\}$, $q_i = \min\{q_i : A_{iq_i}A_{i,q_i+1} \cdots A_{i,c_i-1}A_{i,c_i} \subseteq \pi_i\}$, $T_i^{F^{0\delta_i}} = \beta_i^{\delta_i} + \sum_{l=1, l \neq b_i}^{c_i} \alpha_{il} + \alpha_{ib_i}^0$, $i = 1, \cdots, m - 1$, and $T_m^{F^{\delta_m}} = \beta_m^{\delta_m} + \sum_{l=1}^{c_m} \alpha_{ml}$. If $\exists i$ such that $b_i \notin \Lambda_i^P$, then the term $RCT^0(P_{ib_i})$ in K_{ij} becomes $\max\Big\{RCT^0(Q) : Q \in \big\{P_{ik_L(P_{ib_i})}, P_{ik_R(P_{ib_i})}, R_i\big\}\Big\}$, where $P_{ik_L(P_{ib_i})}$ and $P_{ik_R(P_{ib_i})}$ have the similar definition as in Theorem 2.

For a 2-cluster tool $\mathbb{C}_1$ and $\mathbb{C}_2$, when the buffer is double-space, there could be three possible cycle times that are given in the following theorem.

Theorem 3: For a 2-cluster tool consisting of single-cluster tools $\mathbb{C}_1$ and $\mathbb{C}_2$ connected by a double-space buffer P_{1b_1} and under schedule $\boldsymbol{\pi} = (\pi_1, \pi_2)$, the three possible cycle times are, respectively,

(i) $T_{1,2}(\pi_1, \pi_2; n_1 - 1, n_2)$ is the same in Theorem 2.
(ii) $T_{1,2}(\pi_1, \pi_2; n_1, n_2) = \max\{T_1^0(\pi_1; n_1), T_2^0(\pi_2)\}$;

(iii)

$$T_{1,2}(\pi_2, \pi_2; n_1+1, n_2) = \max\{T_1^0(\pi_1; n_1), T_2^0(\pi_2), RR_{12}\}$$

where $RR_{12} = RCT(R_1) + RCT(R_2) + 2(\epsilon_1 + \epsilon_2) - (RCT^0(P_{1b_1}) + RCT^0(P_{20}))$.

From Theorem 3, we conclude that given a schedule $\pi = (\pi_1, \pi_2)$, the optimal job distribution is (n_1, n_2) if the buffer is double-space and $(n_1 - 1, n_2)$ if the buffer is single-space, where n_1 and n_2 are determined by Lemma 1. One main reason for deriving the closed form expression for the minimal cycle time is that the closed form expression reveals the structural properties of the cycle time, and thus allows us to develop efficient algorithms to finding the optimal schedule. The following theorem states that the optimal schedules for a 2-cluster tool can be found in polynomial time. The results derived for scheduling branch-cluster tools can be extended to find the smallest cycle time of an M-cluster tool [8].

Theorem 4: Finding an optimal schedule for a 2-cluster tool can be solved in polynomial time.

IV. Efficient Scheduling of Multi-Cluster Tools

In this section, we develop conditions under which a multi-cluster tool can be decoupled such that existing efficient algorithms for single-cluster tools can be used to find the optimal schedules. We also derive optimality conditions under which the pull schedule is optimal.

Lemma 4: For a 2-cluster tool with a single-space buffer, the two dependency terms $RCT^0(P_{1b_1}) + t_{S_1}$ and K_{12} in Theorem 2 are minimized under the pull schedule (π_1^p, π_2^p),

$$(\pi_1^p, \pi_2^p) = \underset{(\pi_1,\pi_2)}{\arg\min}\{RCT^0(P_{1b_1}) + t_{S_1}\}, \quad (3a)$$

$$(\pi_1^p, \pi_2^p) = \underset{(\pi_1,\pi_2)}{\arg\min}\{K_{12}\}. \quad (3b)$$

We define the concept of *decoupling equivalence (DE)* is a property of an M-multi-cluster tool with which the throughput of the tool is equal to the maximum throughput of the M decoupled single-cluster tools,

$$T_{1,\dots,M}(\pi_1, \dots, \pi_M) = \max\left\{\max_{i\in\Omega_D\cup\mathbb{L}} T_i^{\mathbf{0}}(\pi_i), \max_{i\in\Omega_S} T_i^{\mathbf{ts}}(\pi_i)\right\},$$

where $T_i^{\mathbf{0}}(\pi_i) = T_i(\pi)|_{t_{ib_{ij}}=0}$, $j \in \Phi_i^D$, $i \in \Omega_D$, $T_i^{\mathbf{0}}(\pi_i) = T_i(\pi)$, $i \in \mathbb{L}$ (index set of leaf clusters), $T_i^{\mathbf{ts}}(\pi_i) = T_i(\pi)|_{t_{ib_{ij}}=t^d_{S_{\rho(ib_{ij})}}}$, $j \in \Phi_i^S$, $i \in \Omega_S$, $t^d_{S_{\rho(b_{ij})}} = 4\epsilon_{\rho(b_{ij})} + 3\delta_{\rho(b_{ij})}$, and Ω_S (Φ_i^S) and Ω_D (Φ_i^D) denote the index sets of clusters (buffers) with a single-space buffer module and a double-space buffer module, respectively.

By Theorem 3, a 2-cluster with double-space buffer modules can be decoupled because it possesses the DE property. However, if the buffer module is single-space, the existence of the DE property depends on the timing data and the robot schedules. We thus consider the DE property under the single-space buffer case. Moreover, we only provide the conditions for $b_i \in \Lambda_i^P$. The conditions for $b_i \in \Lambda_i^R$ can be obtained in a similar fashion.

Proposition 1: Under schedule $\pi = (\pi_1, \pi_2)$, a 2-cluster tool with a single-space buffer module possesses the DE property iff $\max_{i=1,2}\left\{RCT(P_{ij_i^{\max}}), RCT(R_i)\right\} \geq \max\left\{RCT^0(P_{1b_1}) + t_{S_1}, K_{12}\right\}$, where $j_i^{\max} = \arg_{j\neq b_i}\max RCT(P_{ij})$, $i = 1, 2$.

We next focus on the DE property under the pull schedule.

Proposition 2: Under the pull schedule, a 2-cluster tool with a single-space buffer module possesses the DE property iff (1) at least one of the conditions (i) and (ii) is satisfied, and (2) at least one of the conditions (iii) and (iv) is satisfied:

(i) $\max\left\{t_{1j_1^{\max}}, 2(c_1-1)\epsilon_1 + (2c_1-1)\delta_1\right\} \geq 4\epsilon_2 + 3\delta_2$;
(ii) $\max\left\{t_{2j_2^{\max}}, 2(c_2-1)\epsilon_2 + (2c_2-1)\delta_2\right\} \geq 4\epsilon_1 + 3\delta_1$;
(iii) $c_2\max\left\{(c_1+1)\beta_1, \beta_1 + \alpha_{1j_1^{\max}}\right\} \geq \beta_1 + \beta_1^{\delta_1} + \beta_2^{\delta_2} + \sum_{l=1}^{c_2}\alpha_{2l}$; and
(iv) $c_2\max\left\{(c_2+1)\beta_2, \beta_2 + \alpha_{2j_2^{\max}}\right\} \geq \beta_1 + \beta_1^{\delta_1} + \beta_2^{\delta_2} + \sum_{l=1}^{c_2}\alpha_{2l}$.

Roughly speaking, Conditions (i) or (ii) in Proposition 2 imply that to satisfy the DE property, either the time for R_2 to return a job to the buffer is smaller than at least one of the processing times at $\mathbb{C}_1$; or the time for R_1 to return a job to the buffer is smaller than at least one of the processing times (or robot cycle time) at $\mathbb{C}_2$. We can interpret Conditions (iii) or (iv) similarly. Therefore, in order to decouple a 2-cluster tool $\mathbb{C}_1$ and $\mathbb{C}_2$ under the pull strategy, the cluster tool should have one long processing time (or robot cycle time) in either $\mathbb{C}_1$ or $\mathbb{C}_2$, and a short total processing and moving times at $\mathbb{C}_2$.

Proposition 3: If the pull schedule minimizes the cycle time expression in Eq. (3), then the pull schedule is optimal.

Proposition 3 implies that the decomposition approach in [6] indeed leads to optimal schedules because the pull schedule minimizes the decoupling single clusters when the moving time is zero. Moreover, Proposition 3 also generalizes the results given by Theorem 3 in [1] to multi-cluster tools.

In practical applications, the robot moving time among PMs is relatively small. When the robot moving time is smaller than the processing times, namely, $t_{ij} \geq \delta_i$, $j = 1, \dots, c_i$, there is a high incentive for keeping the robots R_i moving rather than waiting at any particular PM for the sake of a shorter cycle time. This intuition has been verified in [1] and summarized as follows.

Proposition 4: The pull schedule (π_1^p, π_2^p) is optimal for a 2-cluster tool when $t_{ij} \geq \delta_i$, $j = 1, \dots, c_i$, $j \neq b_1$, $i = 1, 2$, and $4\epsilon_2 + 3\delta_2 \geq \delta_1$.

V. An Industrial Example

Fig. 3 shows a schematic of a chemical-mechanical planarization (CMP) polisher used in semiconductor manufacturing. CMP is widely used to planarize the wafer surface and to enhance the photolithograph process performance. The CMP polisher can be modeled as a 4-cluster tool with four single-blade robots R_i, $i = 1, \dots, 4$. A similar system was studied in [5] under an assumption of zero robot moving times. Here we relax such an assumption. The wafer processing times are shown in Table II. The wafers pass

TABLE I

CYCLE TIME CALCULATION FOR THE CMP POLISHER

Time	$T_1^0(\pi_1)$	$T_2^{tS_2}(\pi_2)$	$T_3^{tS_3}(\pi_3)$	T_4^0	$RCT(P_{23})$	$RCT(P_{31})$	K_{23}	K_{34}	K_{24}	$T_{1234}(\pi)$
s	35	125	11	64	45	11	45	65	77	125

through the cluster tool as the following flow chart:

$$C_1 \xrightarrow{R_1} P_{11} \xrightarrow{R_2} P_{21} \xrightarrow{R_2} P_{22} \xrightarrow{R_2} P_{23} \xrightarrow{R_3} P_{31} \xrightarrow{R_4} P_{41}$$
$$\xrightarrow{R_4} P_{42} \xrightarrow{R_4} P_{43} \xrightarrow{R_4} P_{40} \xrightarrow{R_3} P_{30} \xrightarrow{R_2} P_{24} \xrightarrow{R_1} C_2.$$

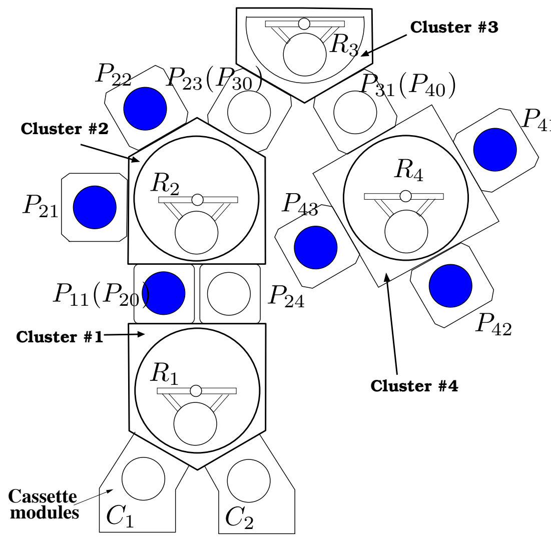

Fig. 3. A schematic of the 4-cluster CMP polisher layout.

From Corollary 1, under schedule $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \pi_4)$, we obtain the cycle time of the polisher as

$$T_{1234}(\pi_1, \pi_2, \pi_3, \pi_4) = \max\left\{T_1^0(\pi_1), T_{234}(\pi_2, \pi_3, \pi_4)\right\} =$$
$$=\max\{T_1^0(\pi_1), T_2^{tS_2}(\pi_2), T_3^{tS_3}(\pi_3), T_4^0(\pi_4), K_{23}, K_{34}, K_{24}\},$$

The optimality conditions specified in Proposition 4 are easily verified. The robot pull schedules are optimal and the resource cycle times are listed in Table I. The final one-unit cycle time for the polisher is 125 s. For the given data, the n-unit job-resource cycle (here $n = 3$) and the two buffer resource cycles do not dominate the cycle time, and this cluster tool possesses the DE property.

TABLE II

CLUSTER TOOL TIMING PARAMETERS

Clusters	ϵ_i (s)	δ_i (s)	t_{ij} (s)
$\mathbb{C}_1$	5	5	N/A
$\mathbb{C}_1$	5	5	$t_{21} = 90$, $t_{22} = 30$
$\mathbb{C}_1$	1	1	N/A
$\mathbb{C}_1$	1	0	$t_{41} = t_{42} = t_{43} = 60$

We perform a large number of Monte Carlo simulations to examine the effect of the uncertainties in processing, loading/unloading, and moving times to the DE property and the optimality of the pull schedule. Each simulation includes 1,000,000 replications, all following the same distribution. In all experiments, $\epsilon_i, \delta_i \sim U(0.1, 10)$, $i = 1, 2$ and

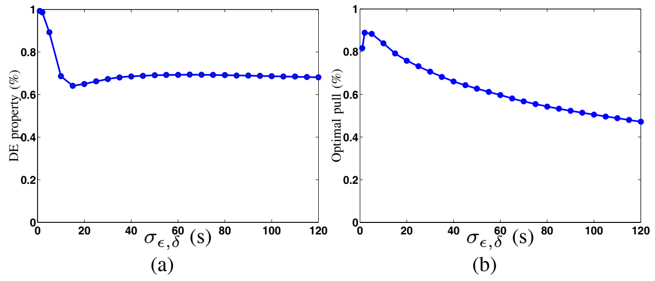

Fig. 4. Sensitivity analyses of DE property and pull optimality versus the variation of robot moving and load/unload time.

$\epsilon_i, \delta_i \sim U(0.1, 1)$, $i = 3, 4$. Fig. 4 shows that when both the robot moving and load/unload times change, the average percentages that the DE property holds and the pull schedule is optimal decrease quickly at the beginning but very slowly at around 68% and 47%, respectively.

VI. CONCLUSION

This paper presented the k-unit optimal scheduling problem for multi-cluster tools with single-blade robots. The inter-cluster interactions among multiple clusters create the phenomena of k-unit job-resource cycles. We used a resource-based method to analytically capture the dependencies among clusters. A closed-form cycle time expression was obtained for multi-cluster tools and a polynomial-time algorithm to finding the optimal schedule was also provided for 2-cluster tools. Decoupling conditions and optimal conditions of the robot pull strategy were also established for a multi-cluster tool. A CMP polisher in semiconductor manufacturing production is used as an example to illustrate the proposed formulation and algorithms.

REFERENCES

[1] M. Dawande, C. Sriskandarajah, and S. Sethi, “On throughtput maximization in constant travel-time robotic cells,” *Manuf. & Serv. Oper. Manag.*, vol. 4, no. 4, pp. 296–312, 2002.

[2] N. Geismar, M. Dawande, and C. Sriskandarajah, “Approximation algorithms for k-unit cyclic solutions in robotic cells,” *Europ. J. Oper. Res.*, vol. 162, pp. 291–309, 2005.

[3] S. Kuma, N. Ramanan, and C. Sriskandarajah, “Minimizing cycle time in large robotics cell,” *IIE Trans.*, vol. 37, no. 2, pp. 123–136, 2005.

[4] N. Geismar, C. Sriskandarajah, and N. Ramanan, “Increasing throughput for robotic cells with parallel machines and multiple robots,” *IEEE Trans. Automat. Sci. Eng.*, vol. 1, no. 1, pp. 84–89, 2004.

[5] S. Ding, J. Yi, and M. T. Zhang, “Multi-cluster tools scheduling: an integrated event graph and network model approach,” *IEEE Trans. Semiconduct. Manufact.*, vol. 19, no. 3, pp. 339–351, 2006.

[6] J. Yi, S. Ding, D. Song, and M. T. Zhang, “Steady-state throughput and scheduling analysis of multi-cluster tools: An decomposition approach,” *IEEE Trans. Automat. Sci. Eng.*, vol. 5, no. 2, pp. 321–336, 2008.

[7] W.-K. Chan, J. Yi, and S. Ding, “On the optimality of one-unit cycle scheduling of multi-cluster tools with single-blade robots,” in *Proc. IEEE Int. Conf. Auto. Sci. Eng.*, Scottsdale, AZ, 2007, pp. 392–397.

[8] W.-C. Chan, J. Yi, and S. Ding, “Optimal scheduling of multi-cluster tools with finite inter-cluster buffers and single-blade robots,” 2008, submitted to *Oper. Res.*, also downloadable at http://sigma.dses.rpi.edu/ROSL/Others/MC/.