Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds

A NOVEL SIMULATION METHODOLOGY FOR MODELING CLUSTER TOOLS

Emrah Cimren
Robert Havey
DongJin Kim

Micron Technology Inc.
9600 Godwin Drive
Manassas, VA 20110, USA

ABSTRACT

Cluster tools are highly integrated machines that can perform multiple manufacturing processes. A series of processing steps, transportation, and control are integrated into a single tool. We develop a novel simulation methodology to determine production schedules for cluster tools. The proposed approach provides route of products in the tool with given process times and scheduling rules. Framework and algorithms used in the simulation are presented. Based on the methodology, we develop a simulation model for a scanner cluster tool used in the photolithography process in semiconductor manufacturing. The impact of different input factors on the tool throughput and cycle time is investigated.

1 INTRODUCTION

Cluster tools are highly integrated machines that can perform multiple manufacturing processes. A series of processing steps, transportation, and control are integrated into a single tool (Singer 1995). Cluster tools are widely used in modern manufacturing industry such as printed circuit board electroplating lines and semiconductor manufacturing systems. In these industries, since product life cycle is very short, on time delivery of products to customers is important. On time delivery performance can be improved by reducing manufacturing cycle time by better planning and scheduling of cluster tools (Kim et al. 2012).

A cluster tool consists of processing modules, buffer modules, and transfer modules (robots, carriers etc.). Jobs are the products which are required to be processed on the tool. Each job visits processing modules based on its predetermined route. Processing modules can perform identical, different or both processes. Processing of a job is not allowed to interrupt at any point of time (no preemption). Set up may be required before or after a job is processed. Buffer modules store jobs until the next step on the job's route becomes available. Transfer modules transfer job between processing modules and buffers.

Since a series of processing steps are integrated into one machine, the structure of cluster tools is complex and sophisticated (Lee 2008). Since small changes on an individual module may have significant impact on the overall tool performance, modeling methodologies are required for modeling, analyzing, and improving cluster tools performance. These models can be used to compare existing tool designs with alternatives and evaluate different scheduling policies.

Dummler (2004) presents approaches to model cluster tools; analytical models and discrete event simulation models. Analytical models are closed formulae approaches, Petri Nets, and other models (i.e. optimization and heuristics). Closed formulae approaches provide solutions to subset of cluster tool scheduling problems (Perkinson and Gyurcsik 1996; Venkatesh et al. 1997; Wood 1996; Wu and Zhou 2010; Geismar et al. 2011). However, they have limitations concerning the complexity of tool types considered and the flexibility of the models in terms of changes in configuration, etc.

A cluster tool with buffers, a transfer robot, and process modules can be modeled as a robotic flow shop where buffers and process modules are equivalent to input stations and machines, respectively. The cluster tool sub-problem is to schedule a given jobs on machines to minimize cycle time. Crama and van de Klundert (1997) show that the robotic flow shop problem, which is equivalent to the cluster tool sub-problem, is strongly NP-Complete (Garey and Johnson 1979).

Mathematical programming methods, including linear programming and mixed integer programming, are used to model cluster tools (Bhushan and Karimi 2003; Karimi et al. 2004; Aguirre et al. 2010). They provide more flexibility in modeling cluster tools than closed formula approaches. However, the range and complexity of cluster tool types are still limited. Since the majority of the cluster tools scheduling problems are NP-Complete, obtaining optimal solution of these models requires high computational efforts. Therefore, these models do not provide flexibility in investigating different configuration changes.

If a cluster tool has a complex structure, i.e. multiple robots, then it cannot be analyzed using closed formulae approaches and mathematical programming methods. Petri Nets, which are a graphical and mathematical modeling formalism for designing and analysis of discrete event systems (Kim et al. 2011), are used to model complex cluster tools. Various versions of cluster tools, mostly in semiconductor manufacturing, are modeled using Petri Nets and extended versions of Petri Nets (e.g. Colored Petri Nets) (Jeng and Zhau 1998; Jung and Lee 2008; Lee 2010). The existing Petri Net modeling software tools cannot handle large scale real life models very well. Run time of the large scale models is quite high and this limits flexibility of model use to analyze different what-if scenarios (Vojnar 1997). Applicability of the model depends on the version of the modeling software that model is built in, and this restricts future model updates (Wells 2002). Also, visualization of the model results is limited.

Discrete event simulation models are widely used to model cluster tools. One of the common approaches of simulating cluster tools is to use a general purpose simulation environment which allows to model a wide variety of technical systems and that are not focused on a specific field of application (e.g. FlexSim, ToolSim). Such models provide the user with a tool set of generic components that allow composing a model of a given technical system (Pierce and Drevna 1992, LeBaron and Pool 1994). The models can be used to assess performance of individual tools; however, analysis capability is limited based on tool type and layout. Also, these models cannot be used for performance analysis of a set of cluster tools (Seppanen 1998). Dummler (1999) presents a hybrid method (CluSim) combining simulation and a genetic algorithm heuristic method to develop a cluster tool schedule. In the method, first a set of candidate schedules are determined using the genetic algorithm heuristic and simulation. Then, these schedules are evaluated with simulation to finally determine the best schedule among the candidate list. Although this method compares different schedules, limited number of scheduling rules is used. Schruben (1999) presents a simulation model which is based on event-graph where model state, events that cause the state of the model to change, and the causality relationships between these events are considered. Pederson and Trout (2002) and Ding and Yi (2004) develop event-graph simulations for cluster tools in semiconductor manufacturing. Run time of event-graph based simulators can be very high when a complex cluster tool is modeled.

In this study, we develop a generic discrete event simulation framework for modeling various types of cluster tools. The framework can be used to build simulation models to determine production schedules for the tools. The method is based on an object-oriented approach which provides high flexibility to model different properties of the tools for different tool types.

This paper is structured as follows. In Section 2, we first provide general definition of the cluster tool. Then, we define simulation framework and algorithms. In Section 3, we develop a simulation model for a scanner tool used in the photolithography process in semiconductor manufacturing based on the proposed methodology. In Section 4, we discuss our results and indicate possible extensions to our approach.

2 METHODOLOGY

In this section, we first provide cluster tool definition and then explain simulation framework.

2.1 Cluster Tool Definition

Cluster tools consist of resources including load ports, transporters, buffers, and process modules. Jobs are loaded to the tool through load ports. Supporting jobs are the jobs which are required to be performed to be able to process a job. For example, running a setup on a process module before processing a job is a supporting job. Process modules are responsible for performing the actual material processing. Buffers are the places where a job can wait until the next process module is available. Transporters carry jobs within the cluster tool to the individual process modules and buffers. The cluster tool scheduling problem is to determine a schedule for the cluster tool with minimum lot cycle time.

We now provide notations to define the problem. Let *N* be the set of jobs and *M* be the set of cluster tool modules including process modules, buffers, and transporters. Let $\overline{\mathrm{N}}$ be a set of supporting jobs. Since a buffer is a processing module with zero processing time, we call buffers as processing modules. Let P be the set of processing modules and T be the set of transporters. Note that $M = P \cup T$. Let T_{i} be set of transporters which can pick or deliver a job to the processing module $i \in P$ where $T = \{T_i | i \in P\}$. Let S_j be the sequence of processing modules that are followed by job $j \in N$. Let S_{jr} be the rth element of the sequence S_j where $S_j = \{S_{j1}, \ldots, S_{jr}\}$ for $j \in N$. For example, S_1 is a processing route of job 1 where $S_1 = \{1,2:3,4\}, S_{11} = \{1\}, S_{12} = \{2:3\}$, and $S_{14} = \{4\}$. In the route, job 1 follows processing module 1, any one of processing modules 2 and 3, and processing module 4. Let E_i be the set of operations that can be performed by module $i \in M$. For example, an operation set of a process may include set up and process; and an operation set of a transporter may include picking a job from process module i_1 and delivering the job to process module i_2 where $\mathrm{i}_1, \mathrm{i}_2 \in \mathrm{P}$. Let p_{ij} and s_{ij} be the processing and setup times of job $j \in N$ on module $i \in M$, respectively. Transportation time is defined as processing time. Let c_i be the capacity of module $\mathrm{i} \in \mathrm{M}$ where $c_i = 1$ for $i \in P$. Process modules which can process more than one job at a time can be modeled as parallel modules with single job capacity. For transporters which can hold more than one job, $c_i \geq 1$ for $i \in T$, we assume that one job can be delivered at a time. Let x_{ij} be the start time of job $j \in N$ on module $i \in M$ and y_{ij} be the end time of job $j \in N$ on module $i \in M \cup \{0\}$. Note that x_{ij} and y_{ij} are calculated as a result of the analysis. Note that y_{0j} be the arrival time of job $j \in N$ to the cluster tool.

2.2 Simulation Framework

Proposed framework consists of four main parts; simulation inputs, simulation algorithm, dynamic entities, and output (Figure 1).

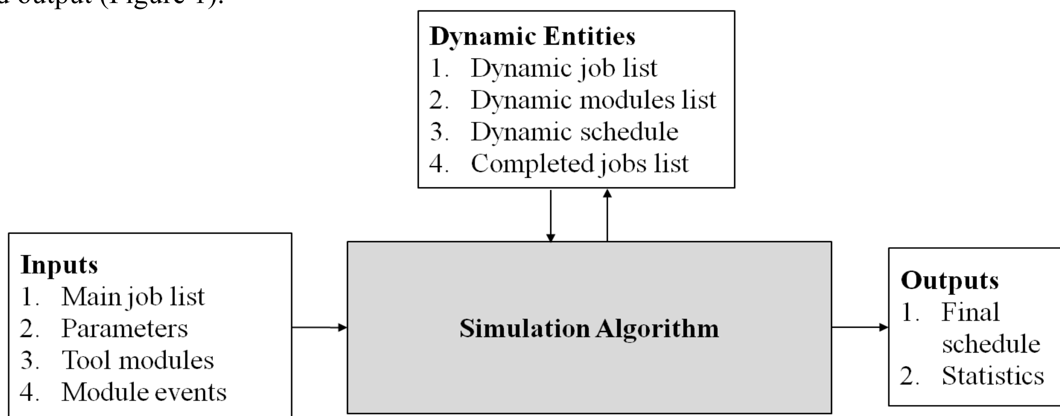


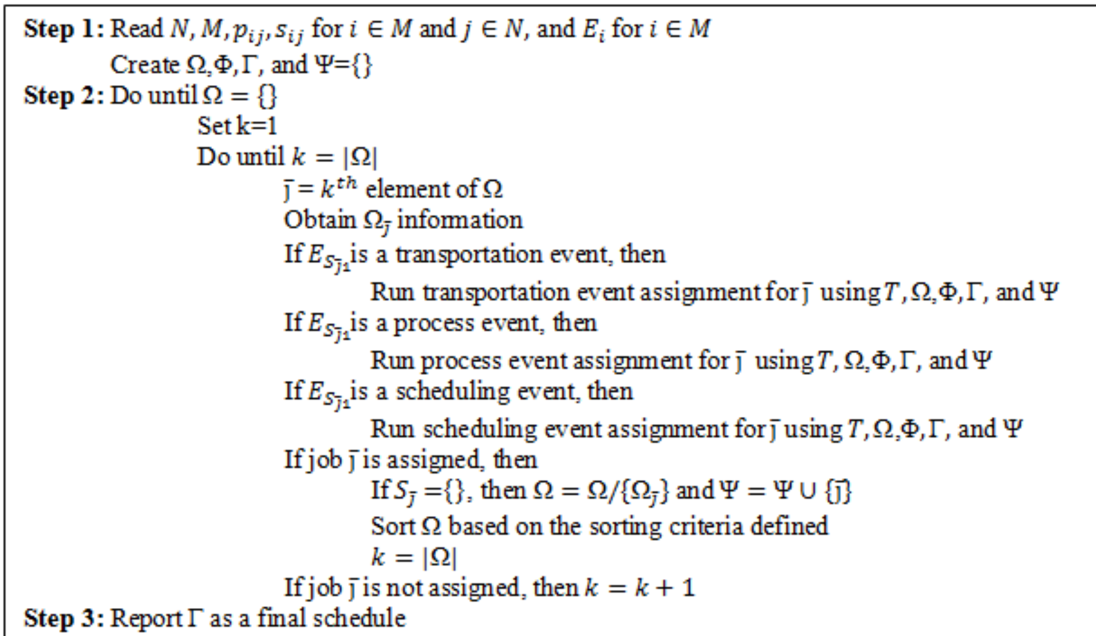
Figure 1: Simulation framework

Four different types of inputs are used in the simulation framework. A main job list (set *N*) includes the jobs processed at the cluster tool. Parameters include processing and setup time, p_{ij} and s_{ij} of job

$j \in N$ on module $i \in M$. Tool modules is a module set M which are the modules in the cluster tool. Module events are all sets E_i which are the set of operations that can be performed by module $i \in M$.

We now describe dynamic entities and then explain simulation algorithm. Dynamic job list keeps information about the current status of each job in the tool (including supporting jobs if required) such as job name, current location of job in the tool, list of modules remained in its processing route, next event required for the first element of route, and last event finish time. Dynamic jobs list is updated when simulation runs with the new information. Let Ω be a set (called dynamic job list) where $\Omega = \{\Omega_j | j \in N\}$ and $\Omega_j = \{j, i, S_j, E_{S_{j1}}, y_{ij}\}$ for $i \in M$ and $j \in N \cup \bar{N}$. Dynamic modules list keeps information about the current status of each module such as module name, name of a job in the module, and last event finish time. Let Φ be a set (dynamic modules list) where $\Phi = \{\Phi_i | i \in M\}$ and $\Phi_i = \{i, j, y_{ij}\}$ for $i \in M$ and $j \in N$. Dynamic schedule includes event information as job name, module name, event name, event start time, event finish time. Let K be the set of events occurred before. Let Γ be a set (dynamic schedule) where $\Gamma = \{\Gamma_k | k \in K\}$. For $k \in K$, $\Gamma_k = \{i_1, j_1, E_{S_{j_{1,r}}}, x_{i_1 j_1}, y_{i_1 j_1}\}$ where $i_1 \in M, j_1 \in N$, and $S_{j_1 r} \in S_{j_1}$. Finally, completed job list includes set of jobs which complete its processing route in the tool. Let Ψ be a set (completed job list) where $\Psi \subseteq N$.

Simulation algorithm is illustrated in Figure 2. In Step 1, first algorithm reads job and module sets, parameters, and event sets. Then, it creates dynamic job list set, dynamic modules set, dynamic schedule set, and completed job list set. Note that completed job list set is empty at the beginning. Step 2 recursively executes assignment process until there is no job left in the dynamic job list set. In this step, at each iteration, dynamic job list is sorted based on the selected criteria (e.g. earliest available job). The first job on the dynamic job list is considered. Algorithm executes the first step in jobs available route list and updates all sets using event start and end times and a new location of the job. If the job is assigned which means either get transferred from one module to another through transporter or get processed on one of the processing modules, then the job's new location and event end time is updated in the dynamic job list set. Also, job and time information in its new location is updated in the dynamic modules set.



Step 1: Read N, M, p_{ij}, s_{ij} for $i \in M$ and $j \in N$, and E_i for $i \in M$
Create Ω, Φ, Γ, and Ψ={}
Step 2: Do until $\Omega = \{\}$
Set k=1
Do until $k = |\Omega|$
$\bar{j} = k^{th}$ element of Ω
Obtain $\Omega_{\bar{j}}$ information
If $E_{S_{\bar{j}1}}$ is a transportation event, then
Run transportation event assignment for $\bar{j}$ using T, Ω, Φ, Γ, and Ψ
If $E_{S_{\bar{j}1}}$ is a process event, then
Run process event assignment for $\bar{j}$ using T, Ω, Φ, Γ, and Ψ
If $E_{S_{\bar{j}1}}$ is a scheduling event, then
Run scheduling event assignment for $\bar{j}$ using T, Ω, Φ, Γ, and Ψ
If job $\bar{j}$ is assigned, then
If $S_{\bar{j}}$ ={}, then $\Omega = \Omega / \{\Omega_{\bar{j}}\}$ and $\Psi = \Psi \cup \{\bar{j}\}$
Sort Ω based on the sorting criteria defined
$k = |\Omega|$
If job $\bar{j}$ is not assigned, then $k = k + 1$
Step 3: Report Γ as a final schedule

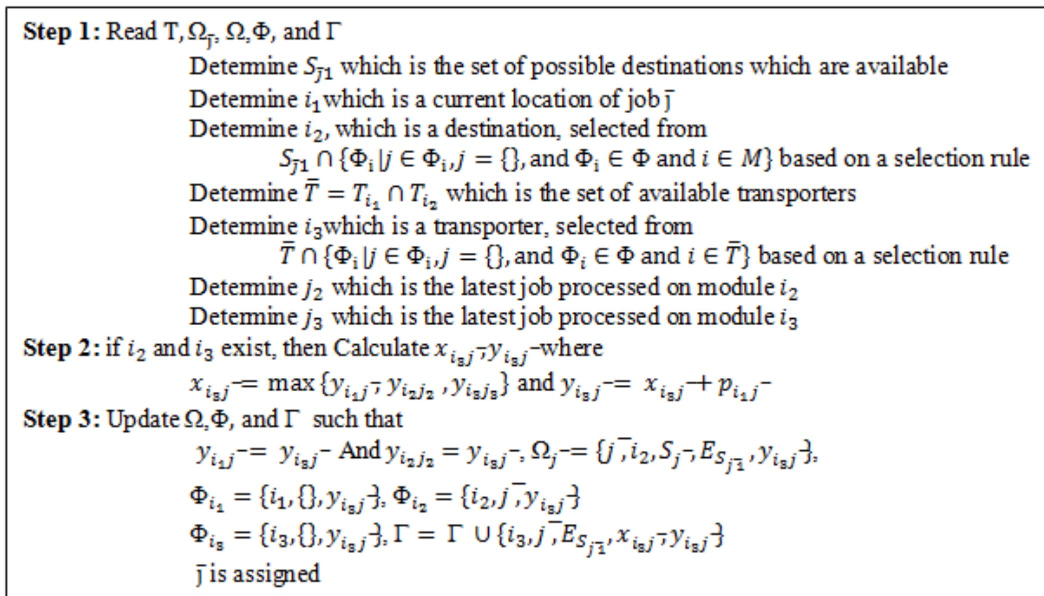
Figure 2: Simulation algorithm

Dynamic schedule set captures job events start and end times and resource information which performs this event. If the job is not assigned, which means either there is no resource available at that time if it is a

transportation event or processing module is not available at that time if it is a process event, then algorithm considers the next job on the list. Step 3 reports the final tool schedule.

Figure 3 illustrates the "transportation event assignment" algorithm which can be executed depend on the current job event in Step 2 of the simulation algorithm. In transportation event assignment, the current job $\bar{j}$ is transported from the current location to a destination through a transporter. Algorithm determines set of possible destinations which are the first element of the route set are. A destination is selected from possible destinations which are empty based on a selection rule (i.e. first available destination). Set of available transporters are identified from an intersection of the set of transporters which are serving to the current location and the set of transporters which are serving to the destination location.

A transporter is selected among the available transporters based on a selection rule (i.e. first available transporter). The latest jobs performed on the destination location and the transporter are identified. In Step 2, if there exists an available destination and transporter, then the current job start time is the latest available time of the current job, destination, and transporter. End time of the current job is the start time of the job and the transfer time. In Step 3, for the current job, new destination, next event, and job available time are updated in the dynamic job list. Job is removed from the dynamic module list of the current location and added to the destination location. For the current and destination locations, and transporter, the available time is set to event end time of the current job. The dynamic schedule set is updated with the job, transporter module, event, event start, and event end information. The status of job is reported as "assigned" for the simulation algorithm.


Step 1: Read $T, \Omega_{\bar{j}}, \Omega, \Phi$, and Γ

Determine $S_{\bar{j}1}$ which is the set of possible destinations which are available

Determine i_1 which is a current location of job $\bar{j}$

Determine i_2, which is a destination, selected from

$S_{\bar{j}1} \cap \{\Phi_i | j \in \Phi_i, j = \{\}$, and $\Phi_i \in \Phi$ and $i \in M\}$ based on a selection rule

Determine $\bar{T} = T_{i_1} \cap T_{i_2}$ which is the set of available transporters

Determine i_3 which is a transporter, selected from

$\bar{T} \cap \{\Phi_i | j \in \Phi_i, j = \{\}$, and $\Phi_i \in \Phi$ and $i \in \bar{T}\}$ based on a selection rule

Determine j_2 which is the latest job processed on module i_2

Determine j_3 which is the latest job processed on module i_3

Step 2: if i_2 and i_3 exist, then Calculate $x_{i_3\bar{j}}, y_{i_3\bar{j}}$ where

$x_{i_3\bar{j}} = \max\{y_{i_1\bar{j}}, y_{i_2j_2}, y_{i_3j_3}\}$ and $y_{i_3\bar{j}} = x_{i_3\bar{j}} + p_{i_1\bar{j}}$

Step 3: Update Ω, Φ, and Γ such that

$y_{i_1\bar{j}} = y_{i_3\bar{j}}$ And $y_{i_2j_2} = y_{i_3\bar{j}}$, $\Omega_{\bar{j}} = \{\bar{j}, i_2, S_{\bar{j}}, E_{S_{\bar{j}1}}, y_{i_3\bar{j}}\}$,

$\Phi_{i_1} = \{i_1, \{\}, y_{i_3\bar{j}}\}$, $\Phi_{i_2} = \{i_2, \bar{j}, y_{i_3\bar{j}}\}$

$\Phi_{i_3} = \{i_3, \{\}, y_{i_3\bar{j}}\}$, $\Gamma = \Gamma \cup \{i_3, \bar{j}, E_{S_{\bar{j}1}}, x_{i_3\bar{j}}, y_{i_3\bar{j}}\}$

$\bar{j}$ is assigned

Figure 3: Transportation event assignment algorithm

Figure 4 illustrates the "process event assignment" algorithm. In the algorithm, the current job $\bar{j}$ is processed based on the first event on its route. Step 1 reads job and module sets, and identifies the current location of job $\bar{j}$. In Step 2, start time of the processing event on the current location for the current event is the available time for the job. Finish time of the processing event is equal to the start time of the event with the process time added. The dynamic schedule set is updated with the job, process module, event, event start, and event end information in Step 3. Route set is updated by removing the process event from the job route. Dynamic job set for the current job is updated with the new sequence, first event in the sequence, and the event end time. The status of job is reported as "assigned" for the simulation algorithm.

The "scheduling event assignment" algorithm is used to model different scheduling rules. These rules are mostly based on the structure and working logic of the tool which is modeled. For example, a scheduling rule may be to run a process set up based on the process condition after job is removed from the processing module. In this case, when job is removed from the process module, a new supporting job (a

scheduling job) is added to the dynamic job list and this job is being executed in the scheduling algorithm based on its order. In this case, the "scheduling event assignment" algorithm is similar to the process event assignment algorithm where processing times are equal to set up times.

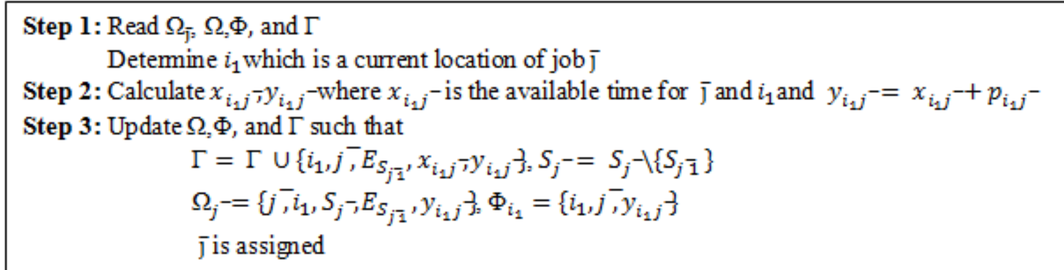


Figure 4: Process event assignment algorithm

3 MODELING PHOTOLITOGRAPHY SCANNER TOOL

In semiconductor manufacturing, chips are produced by a multi-step processing of silicon discs, called "wafers". A part type undergoes about one thousand complex processing steps, including photolithography, etching, chemical and physical vapor deposition, cleaning, and thermal processing. Among these processes, photolithography is one of the critical processes since it constitutes about 11% of total time.

Based on the methodology presented above, we develop a simulation model for a scanner tool used in the photolithography process in semiconductor manufacturing. We present a description of the photolithography scanner tool in the next section. Section 3.2 provides simulation study of this cluster tool.

3.1 Description of the Cluster Tool

The scanner tool has three zones that perform three unique operations, the Reticle Zone, the Transfer Zone and the Process Zone. Table 1 details each tool component and description for each zone, and Figure 5 shows a graphical representation of the scanner tool. The Reticle Zone moves reticles from RLP or IRL to the IRIS and LE modules via the RHR. The Transfer Zone moves wafers from PA into the Process Zone via the LR and ULR. The Process Zone is the area where wafers have an image printed on the wafer by the Lens with a reticle.

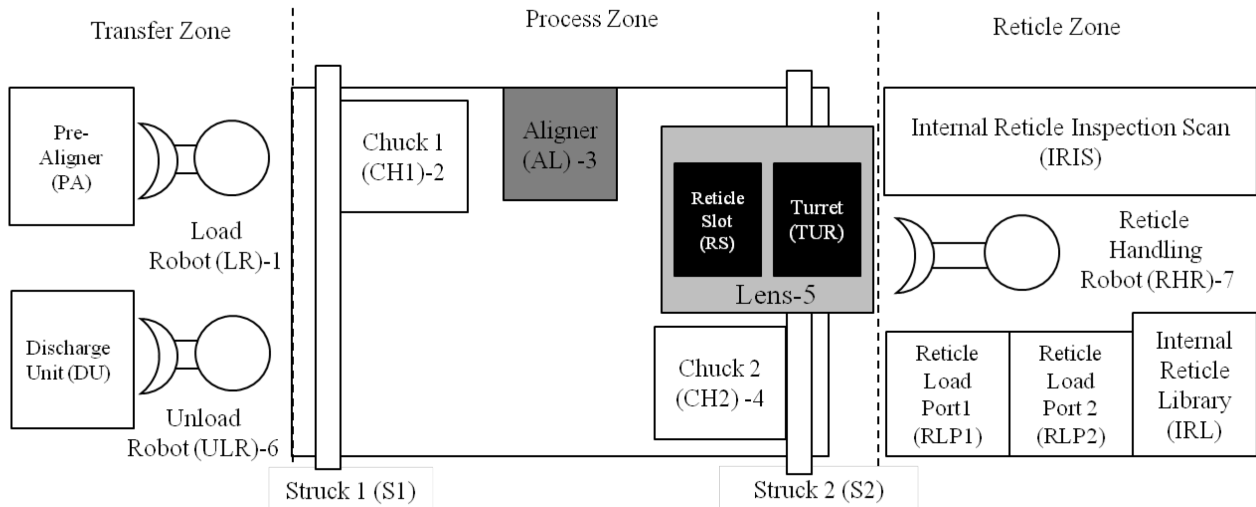


Figure 5: Scanner tool

<u>Reticle Delivery and Reticle Delivery Rules:</u> For a wafer to be processed, a reticle must be present and available for use on the Scanner. A reticle can either be stored in the IRL or delivered to the tool via a Reticle POD to a RLP. Before a reticle is available for use, it must be inspected by the IRIS module to

ensure the reticle is free of defects. If the reticle is not free of defects, reticle will be rejected and lot requiring reticle will not be started by cluster tool. If reticle is free of defects, lot requiring reticle is allowed to start processing by cluster tool.

Table 1: Scanner Tool Components

Area	Tool Component	Tool Description
Reticle Zone	Reticle Load Port (RLP) 1	Reticle Loadport 1 that holds a POD with up to 6 reticles
	Reticle Load Port (RLP) 2	Reticle Loadport 2 that holds a POD with up to 6 reticles
	Internal Reticle Library (IRL)	Internal reticle storage location for up to 12 reticles
	Internal Reticle Inspection (IRIS)	Modules that inspects reticles from RLP 1, RLP 2 and IRL
Transfer Zone	Pre-Aligner (PA)	Entry point into Scanner
	Load Robot (LR) – 1	Robot removes wafer from PA and places on Chuck
	Unload Robot (ULR) – 6	Robot removes wafers from Chuck and Places on DU
	Discharge Unit (DU)	Exit point from Scanner back into Track
Process Zone	Chuck 1 (CH1) – 2	One of two Chucks that wafer is placed on
	Chuck 2 (CH2) – 4	One of two Chucks that wafer is placed on
	Struck 1 (S1)	Robot arm that moves wither CH1 or CH2 between LR, Aligner and ULR
	Struck 2 (S2)	Robot arm that moves wither Chuck 1 or Chuck 2 to Lens position
	Aligner (AL)	Module that aligns wafer on a chuck for processing by Lens
	Lens (LE) – 5	Module that prints an image on wafer using Reticle and light
	Reticle Slot (RS)	Holds Reticle that Lens is currently using to print image on wafer
	Turret (TUR)	Holds Reticle that Lens will use next to print image on wafer

The three reticle flows, IRIS inspection reticle flow path, reticle move to lens flow path, and removal of reticle from IRL flow path, are the different types of reticle movement that can occur in the Reticle Zone. In IRIS inspection reticle flow path, RHR removes reticle from RLP and moves reticle to IRIS module. IRIS module inspects reticle for defects. After inspection, RHR removes reticle from IRIS module and returns reticle to POD on RLP. In reticle move to lens flow path, RHR removes reticle from RLP or from IRL and moves reticle to TUR. TUR accepts reticle and waits with reticle until RS requires reticle. When RS needs next reticle, TUR and RS will swap reticles. Reticle now on TUR is picked up by RHR and returned to POD on RLP or back to IRL. In removal of reticle from IRL flow path, RHR removes reticle from IRL and places reticle into empty slot in POD on RLP. Reticle is then moved to another Scanner's RLP or placed into storage.

Wafer Delivery and Wafer Delivery Rules: After the reticle is available on tool and inspected, the wafers can start moving into the cluster tool for processing. All wafers follow the same flow path through the scanner. First, LR removes wafer from PA and places wafer on Chuck held by S1. Then, S1 moves chuck to AL Module for wafer alignment. After alignment and image printing by Lens is complete, S1 and S2 move together and swap chucks. S2 moves Chuck to Lens for printing image on wafer. S1 moves Chuck to URL and URL removes wafer from Chuck and places wafer on DU. S1 now moves empty Chuck to LR for next wafer.

Single vs. Double Exposure: Most Masking Levels require a single exposure to print the features (lines and holes) on the wafer. However, some processed, which have such narrow dimensions, requires a double exposure to print the feature. Single exposure uses one reticle; however double exposure requiring two reticles to print the image on the wafer. When printing a double exposure, a reticle swap occurs while the wafer is under the Lens. During the double exposure process, reticles are in positions RS and TUR on the Lens. A wafer will have an image printed from reticle 1 first, RS and TUR will swap reticles, then reticle 2 will print an image on the wafer next. For the next wafer, reticle 2 will print image first, RS and TUR will swap reticles, allowing reticle 1 to print image second. This reticle swap sequence will repeat for the remaining wafers.

Simulation model is implemented in "R" environment (R 2013). Simulation inputs are provided from a MySQL database (MySQL 2013).

3.2 Cluster Tool's Performance Analysis

We consider two lots with five wafers each as an input to the cluster tool in the simulation runs. Thus, $N = \{1, \ldots, 10\}$. We assume that both lots are available at the beginning and required to be processed in order. Module name and module number mapping is provided in Figure 5 (e.g. LR–2 means that module number for LR is 2). Thus, $M = \{1, 2, \ldots, 7\}$. Let c_k be the chuck swap time between wafers $k-1$ and k. Tool runs a set up process (called lot correction) before processing the first wafer in each lot which is included in these wafers exposure times. All reticles are placed in IRL at the beginning and already scanned by IRIS. Let p_7 be the reticle transfer time to TUR for all reticles. The LR robot puts an unprocessed wafer onto the first available chuck. Table 1 illustrates the events associated with each module in the simulation. We combine events in Gantt in calculations for the simplicity.

Using the simulation model, we analyze two main cases; back-to-back two lots with the same recipe (Case 1), back-to-back lots with different recipes (Case 2). We run the simulation model for different scenarios for each case and report makespan which is the leave time of the second lot from the tool.

We consider two recipes, R1 and R2. The recipe R1 requires single exposure and R2 requires double exposure where single exposure time for both R1 and R2 is the same. Total exposure time of R2 is twice as big as the total exposure time of R1. Both recipes use the same align strategy where align time is the half of the single exposure time. Chuck swap time, reticle swap time, and robot transfer times are smaller than the exposure times.

Table 1: Event set

Module	Event Name in Gantt	Event Name in Calculations
LR	Pick wafer from PA, Put wafer to CH, Move CH to LR	Load
CH1, CH2	Swap CH	Chuck swap
ALI	Move CH to ALI, Align	Align
LE	Lot correction, Exposure	Exposure
TUR, RS	Swap reticles	Reticle swap
ULR	Move CH to ULR, Pick wafer from CH, Put wafer to DU	Unload
RHR	Move reticle to TUR, Move reticle to IRL from TUR	Reticle load/unload

Case 1, Back-to-Back Two Lots with the Same Recipe: In Case 1, we consider two lots running recipe R1 using a reticle (called ret1). Figure 6 illustrates a schedule for Case 1. In the schedule, as soon as ret1 is loaded to RS, LR pulls the first wafer from PA. Figure 6 shows that LE is the bottleneck resource where each wafer is required to be processed. Thus, we can write the following theorem which provides an optimal solution for Case 1 for n wafers with the same recipe. Note that if a recipe requires a double exposure, exposure time can be defined as summation of two exposures and a reticle swap time. Thus, the following results hold for the double exposure recipes.

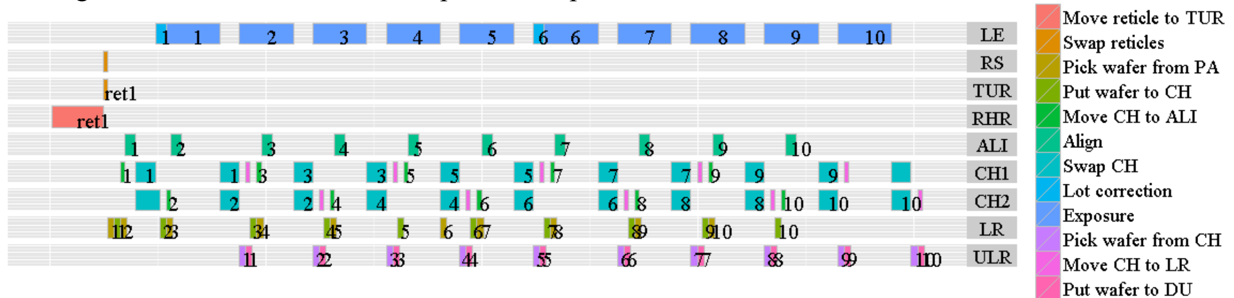


Figure 6: Case 1 schedule

Theorem 1 *Let C_1^* be the optimal makespan for Case1 for N={1,…,n}. Then, C_1^* is calculated as*

$x_{51} = p_7 + p_{11} + p_{31} + c_1,$

$x_{52} = x_{51} + \max\{p_{51}, p_{12} + p_{32}\} + c_2,$

$x_{5k} = x_{5,k-1} + \max\{p_{5,k-1}, p_{6,k-2} + p_{1,k} + p_{3,k}\} + c_k$ for $k = 3, \ldots, n$, and

$C_1^* = x_{5n} + p_{5n} + c_n + p_{6n}.$

Proof Follows from the fact that wafers are required to be processed in order. ■

Proposition 1 *Schedule given in Figure 6 is optimal.*

Proof Follows from Theorem 1. ■

Case 2, Back-to-Back Two Lots with Different Recipes: In Case 2, we consider two lot each consists of five wafers running different recipes. Table 2 illustrates possible scenarios for this case. Exposure time for R3 is the same as the exposure time for R1, and exposure time for R2 is greater than exposure time for R1. Note that wafer from the second lot is not picked unless required reticles are placed in RS and TUR (for the double exposure). We now investigate schedules for each scenario using the simulation model.

Table 2: Possible scenarios for Case 2

Scenario	Lot 1 Recipe	Lot 1 Reticles	Lot 2 Recipe	Lot 2 Reticles
Scenario 1	R1: Single exposure	ret1	R2: Single exposure	ret2
Scenario 2	R3: Double exposure	ret3,ret4	R1: Single exposure	ret1
Scenario 3	R1: Single exposure	ret1	R3: Double exposure	ret3,ret4

Figure 7 illustrates a schedule for Scenario 1. In the schedule, first ret1 is loaded. Then, ret2 is loaded during processing of wafers for the first lot. As soon as, last wafer from the first lot is processed, reticle is swapped and second lot starts processing. In the schedule below, reticle swapping occurs during chuck swapping. Thus, we can write the following theorem which provides an optimal solution for Scenario 1 for Case 2 for n wafers.

Theorem 2 *Let C_2^* be the optimal makespan for Scenario 1 for Case2 for N={1,…,n}. Also, let n_1 be the last wafer for the 1st lot and let h_{n_1} be the reticle swapping time after wafer n_1. If reticle load time is less than the total process time of a lot, then Theorem 1 provides C_2^* where $c_{n_1} = \max\{c_{n_1}, h_{n_1}\}$.*

Proof Follows from the fact that wafers are required to be processed in order and second reticle is already loaded to TUR before first lot finishes processing. ■

Proposition 2 *Schedule given in Figure 7 is optimal.*

Proof Follows from Theorem 2. ■

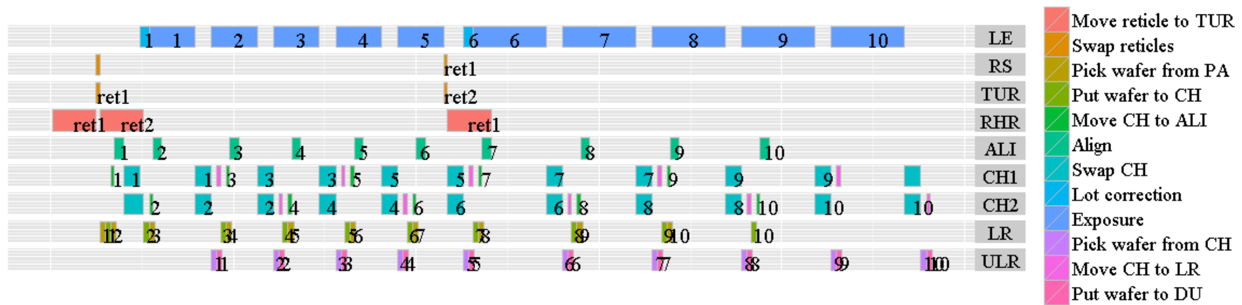


Figure 7: Scenario 1 in Case 2 schedule

Figure 8 illustrates a schedule for Scenario 2 in Case 2. Note that LR puts an unprocessed wafer onto the first available chuck. In the schedule, since R3 is required for the first lot and uses two reticles, ret3 and ret4, both reticles are loaded before process starts. In the double exposure for the first lot, wafer is exposed using both reticles sequentially. When one of the reticles is not needed by the first lot, it is immediately removed from the lens. Second lot waits until ret1 is placed to RS. Since a wafer cannot be pulled by LR from PA until a required reticle is at RS, there is an empty chuck swap after wafer 5 is processed. The following theorem provides an optimal solution for this case.

Theorem 3 *Let C_3^* be the optimal makespan for Scenario 2 in Case2 for N={1,…,n}. Let n_1 be the last wafer for the 1st lot and let h_k be the reticle swapping time after wafer $k \in N$. Then, C_3^* is*

$x_{51} = p_7 + p_{11} + p_{31} + c_1,$

$x_{52} = x_{51} + \max\{2p_{51} + h_1, p_{12} + p_{32}\} + c_2,$

$x_{5k} = x_{5,k-1} + \max\{2p_{5,k-1} + h_{k-1}, p_{6,k-2} + p_{1,k} + p_{3,k}\} + c_k$ for $k = 3, \dots, n_1,$

$x_{5,n_1+1} = x_{5,n_1} + p_{5,n_1} + h_{n_1} + \max\{p_{5,n_1}, p_7\} + \max\{h_{n_1} + 2p_7, c_{n_1}\} + p_{1,n_1+1} + p_{3,n_1+1} + c_{n_1+1}$,

$x_{5k} = x_{5,k-1} + \max\{p_{5,k-1}, p_{6,k-2} + p_{1,k} + p_{3,k}\} + c_k$ for $k = n_1 + 2, \dots, n$, and

$C_3^* = x_{5n} + p_{5n} + c_n + p_{6n}$.

Proof Follows from the fact that wafers are required to be processed in order and both reticles are required to be unload before the second lot starts processing. ■

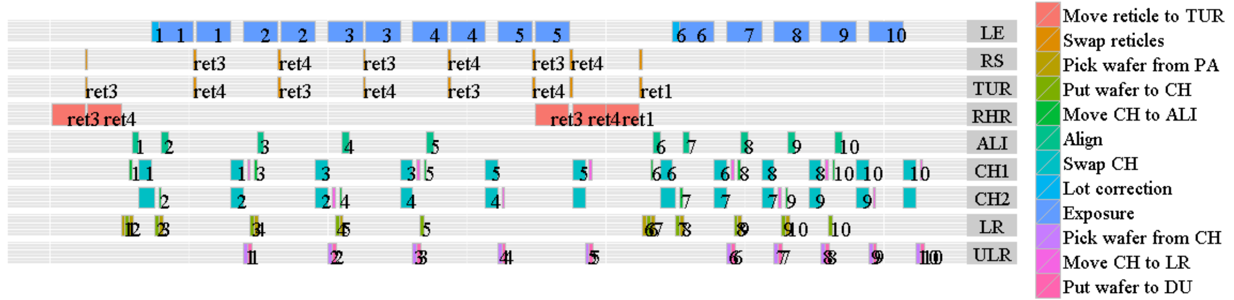


Figure 8: Scenario 2 in Case 2 schedule

Proposition 3 *Schedule given in Figure 8 is optimal.*
Proof Follows from Theorem 3. ■

Suppose that wafers are ordered such a way that both chucks are used in order (i.e wafer 1 to CH1, wafer 2 to CH2,…,wafer 5 to CH1, wafer 6 to CH2,…,wafer 10 to CH2). Figure 9 illustrates a schedule for this case. Note that there is an empty chuck swap event between two lots. Thus, makespan for the schedule in Figure 8 is smaller than for the schedule in Figure 9.

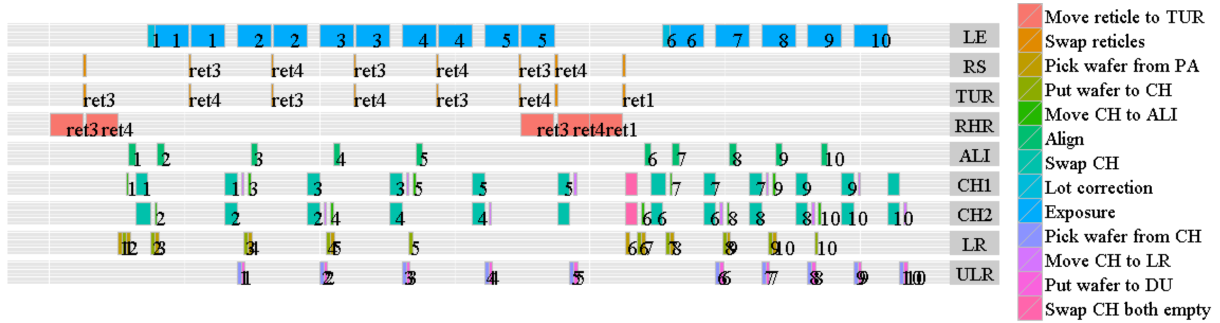


Figure 9: Scenario 2 in Case 2 with pre-assigned chuck order schedule

Figure 10 illustrates a schedule for Scenario 3 in Case 2. In the schedule, since first lots uses only one reticle, ret1 is loaded at the beginning. Since ret3 is required for processing the second lot, ret3 is loaded as soon as ret1 is loaded. After wafer 5 is processed, ret1 is removed from the lens and ret4 is delivered to TUR since ret3 is already in RS. Observer that tool waits until ret4 is loaded to TUR to be able to pull wafer 6 in. The following theorem provides an optimal solution for this case.

Theorem 4 *Let C_4^* be the optimal makespan for Case1 for N={1,…,n}. Let n_1 be the last wafer for the 1st lot and let h_k be the reticle swapping time after wafer $k \in N$. If reticle load time is less than the total process time of a lot, then C_4^* is*

$x_{51} = p_7 + p_{11} + p_{31} + c_1$,

$x_{52} = x_{51} + \max\{p_{51}, p_{12} + p_{32}\} + c_2$,

$x_{5k} = x_{5,k-1} + \max\{p_{5,k-1}, p_{6,k-2} + p_{1,k} + p_{3,k}\} + c_k$ for $k = 3, \dots, n_1$,

$x_{5,n_1+1} = x_{5,n_1} + p_{5,n_1} + \max\{h_{n_1} + 2p_7, c_{n_1}\} + p_{1,n_1+1} + p_{3,n_1+1} + c_{n_1+1}$,

$x_{5k} = x_{5,k-1} + \max\{2p_{5,k-1} + h_{k-1}, p_{6,k-2} + p_{1,k} + p_{3,k}\} + c_k$ for $k = n_1 + 2, \dots, n$,

$C_4^* = x_{5n} + 2p_{5n} + h_n + c_n + p_{6n}$.

Proof Follows from the fact that wafers are required to be processed in order and two reticles are required to load sequentially at the beginning. ■

Proposition 4 *Schedule given in Figure 10 is optimal.*
Proof Follows from Theorem 4. ■

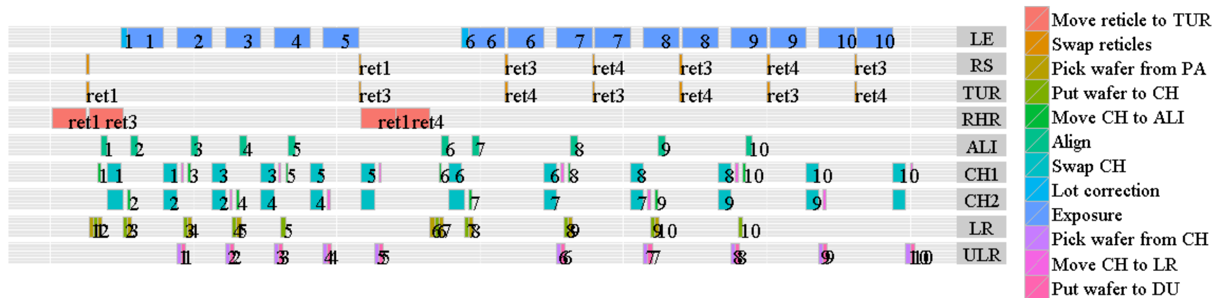


Figure 10: Scenario 3 in Case 2

4 CONCLUSION

A series of processing steps, transportation, and control are integrated in cluster tools. Cluster tools can perform multiple manufacturing processes. Cluster tools are widely used in modern manufacturing industry such as electroplating lines and semiconductor manufacturing systems.

In this study, we present a discrete event simulation methodology to model cluster tools. The proposed approach provides route of products in the tool with given process times and scheduling rules. We provide a literature survey and present limitations of the existing cluster tool modeling methodologies such as closed formulae approaches, Petri Nets, and optimization.

Based on the methodology presented, we develop a simulation model for a scanner tool used in the photolithography process in semiconductor manufacturing. We first provide description of tool elements, tool working logic, and tool rules. Then, we analyze performance of the cluster tool using the model. Two main cases are investigated. Using the model results, we characterize the optimal solution for these cases.

As a future study, we are planning to increase flexibility of the simulation methodology by introducing new structures.

REFERENCES

Aguirre, A.M, C.A. Méndez, P.M. Castro. 2011. “A Novel Optimization Method to Automated Wet-Etch Station Scheduling in Semiconductor Manufacturing Systems.” *Computers & Chemical Engineering* 35(12): 2960-2972.

Bhushan, S. and I.A. Karimi. 2003. “An MILP Approach to Automated Wet-Etch Scheduling.” *Industrial and Engineering Chemistry Research* 42(7): 1391–1399.

Crama, Y. J., v.d. Klundert. 1997. “Robotic Fowshop Scheduling is Strongly NP-Complete”. *Research Memoranda 010, Maastricht, Maastricht Research School of Economics of Technology and Organization*: http://edata.ub.unimaas.nl/wwwedocs/loader/file.asp?id=457.

Ding, S. and J. Yi. 2004. “An Event Graph Based Simulation and Scheduling Analysis of Multi Cluster Tools”. *In Proc. 2004 Winter Simulation Conference*: 1915–1924.

Dummler, M. 1999. “Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance”. *In Proceedings of the 1999 Winter Simulation Conference:* 875-879.

Dummler, M. 2004. “Modeling and Optimization of Cluster Tools in Semiconductor Manufacturing.” Ph. D. Thesis, University of Wurzburg.

Garey, M.R., D.S. Johnson. 1979. "Computers and Intractability: A Guide to the Theory of NP-Completeness". *Freeman*, San Francisco.

Geismar, N., M. Dawande, and C. Sriskandarajah. 2011. "Productivity Improvement From Using Machine Buffers in Dual-Gripper Cluster Tools." *IEEE Transactions on Automation Science and Engineering:* 8(1): 29.

Jeng, M.D, M. Zhou. 1998. "Editorial Introduction to the Special Section on Petri Nets in Semiconductor Manufacturing". *IEEE Transactions on Semiconductor Manufacturing*: 11(3), 330–332.

Jung, C., T.E. Lee. 2008. "Efficient Scheduling Method Based on an Assignment Model for Robotized Cluster Tools". *IEEE International Conference on Automation Science and Engineering*: 79–84.

Karimi, I. A., Y. Zerlinda, Y. L. Tan, and S. Bhushan. 2004. "An Improved Formulation for Scheduling an Automated Wet-Etch Station." *Computers & Chemical Engineering* 29(1): 217-224.

Kim, D.J, E. Cimren, R. Havey, and A. K. Zaidi, 2012. "Improving Cluster Tools Performance Using Colored Petri Nets in Semiconductor Manufacturing." *Winter Simulation Conference* : 1-205.

Lee, T. E. 2008. "A Review of Scheduling Theory and Methods for Semiconductor Manufacturing Cluster Tools." *In Proceedings of the 2008 Winter Simulation Conference,* ed S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler: 2127-2135.

Lee, J., T.E. Lee. 2010. "An Open Scheduling Architecture for Cluster Tools". *In Proc. of 2010 IEEE Conference on Automation Science and Engineering (CASE)*: 420–425.

LeBaron, T. H. and M. Pool. 1994. The Simulation of Cluster Tools: A New Semiconductor Manufacturing Technology." *Winter Simulation Conference*: 907–912.

MySQL. 2013. http://www.mysql.com/.

Pederson, D., and C. Trout. 2002. "Demonstrated Benefits of Cluster Tool Simulation". *In Proceedings of the 2002 Conference on Modeling and Analysis of Semiconductor Manufacturing*: 84–89.

Perkinson, T. L. and R. S. Gyurcsik. 1996. "Single-Wafer Cluster Tool Performance: An Analysis of the Effects of Redundant Chambers and Revisitation Sequences on Throughput." *IEEE Transactions on Semiconductor Manufacturing* 9(3): 384–400.

Pierce, N. G. and M. J. Drevna. 1992. "Development of Generic Simulation Models to Evaluate Wafer Fabrication Cluster Tools." *Winter Simulation Conference*: 874–878.

R. 2013. "The R Project for Statistical Computing": http://www.r-project.org/.

Schruben, L. W. (1999). "Deadlock Detection and Avoidance in Cluster Tools". *In Proceedings of the International Conference on Semiconductor Manufacturing Operational Modeling and Simulation*: 31–35.

Seppanen, M.S. 1998. "Modeling Cluster Tool Configurations in the Wafer Fabrication." *Arena Sphere.*

Singer, P. 1995. "The Driving Forces in Cluster Tool Development." *Semiconductor Int.*: 113–118.

Venkatesh, S., R. Davenport, P. Foxhoven, and J. Nulman. 1997. "A Steady-State Throughput Analysis of Cluster Tools: Dual-Blade Versus Single-Blade Robots." *IEEE Transactions on Semiconductor Manufacturing* 10(4): 418–424.

Vojnar, T. 1997. "Various Kinds of Petri Nets in Simulation and Modelling." *In: Proceedings of 31st Spring International Conference on Modelling and Simulation of Systems MOSIS'97*: 227-232.

Wells L.. 2002. Performance Analysis Using Coloured Petri Nets. Ph.D. Dissertation.

Wood, S. C. 1996. "Simple performance models for integrated processing tools." *IEEE Transactions on Semiconductor Manufacturing* 9(3): 320–328.

Wu, N. and M. Zhou. 2010. "A Closed-Form Solution for Schedulability and Optimal Scheduling of Dual-Arm Cluster Tools With Wafer Residency Time Constraint Based on Steady Schedule Analysis." *Automation Science and Engineering, IEEE Transactions* 7(2): 303-315.

AUTHOR BIOGRAPHIES

EMRAH CIMREN is an industrial engineer in the Industrial Engineering Department at Micron Technology Inc., Virginia. He holds a B.S. degree in Industrial Engineering from Istanbul Technical University in Turkey, an M.S. degree in Industrial Engineering from Sabanci University in Turkey, and a Ph.D.

degree in Industrial and Systems Engineering from The Ohio State University. He is a member of INFORMS. His e-mail is ecimren@ micron.com.

ROBERT HAVEY is an industrial engineer in the Industrial Engineering Department at Micron Technology Inc., Virginia. He received B.S. in Industrial Engineering from Texas A&M University and has done 11 years of semiconductor and Automated Material Handling System planning. His e-mail is rhavey@ micron.com.

DONGJIN KIM is an operations improvement manager in the Industrial Engineering Department at Micron Technology Inc., Virginia. He received M.S. in Engineering Management from Queensland University of Technology, Australia, an MBA from Georgetown University, and has performed 15 years of simulation and modeling experiences in semiconductor industry. He is a member of INFORMS and APICS. His e-mail is djkim@micron.com.