Proceedings of the 2004 Winter Simulation Conference
R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

AN EVENT GRAPH BASED SIMULATION AND SCHEDULING ANALYSIS OF MULTI-CLUSTER TOOLS

Shengwei Ding

Dept. of Industrial Engineering and Operation Research
University of California at Berkeley
Berkeley, CA 94720, U.S.A.

Jingang Yi

Lam Research Corporation
CMP/Cleaning Technology Division
Fremont, CA 94538, U.S.A.

ABSTRACT

Simulation methods are extensively used in modeling complex scheduling problems. However, traditional layout of simulation models can become complicated when they are used to find optimal scheduling in complex systems such as multi-cluster tools for semiconductor manufacturing. In this paper, we study a decision-moving-done method of event driven simulation for multi-cluster tools. Based on this method, we are able to manage all the activities in the simulation. The proposed event graph based simulation study can further be integrated into a pruning search algorithm to find the optimal robot scheduling sequence. Incorporated with simulation model, the search algorithm detects deadlocks and significantly reduces the computing time. A chemical-mechanical planarization (CMP) polisher is used as an example of the multi-cluster cluster tools to illustrate the proposed event graph based simulation and scheduling analysis.

1 INTRODUCTION

Cluster tools are widely used in semiconductor manufacturing. In general, a single-cluster tool consists of one or more cassette and process modules and normally one transfer module (Figure 1(a)). Cassette modules store the unprocessed and processed wafers and process modules execute the semiconductor manufacturing processes, such as deposition, etching and chemical-mechanical planarization. Transfer modules (such as single-blade and double-blade robots) move the wafers among process modules and between process and cassette modules. Since wafers are processed to produce the integrated circuits through multiple sequential process steps, modeling analysis and scheduling of cluster tools is critical to improve the production throughput and enhance the design of processing equipments.

The scheduling of a cluster tool depends greatly on the design and configurations of the tool. For example, if there are two wafers to be picked and placed, the moving sequences for a single-blade transfer robot can only be pick1-place1 and pick2-place2 in series. The scheduling of such a robot configuration is straightforward and can be solved analytically. In contrast, a double-blade robot can use the second blade (or arm) as a buffer and therefore the pick/place sequences can be varying. For the two wafer pick/place example, the robot moving sequences could be pick1-pick2-place1-place2, or pick1-pick2-place2-place1, etc. Consequently, the analytical study of schedules becomes complicated. In Perkinson et al. (1994), Venkatesh et al. (1997), analytical models of steady-state throughput were discussed for a single-cluster tool. To model the processes of cluster tools, Srinivasan (1998) and Zuberek (2001) used Petri nets to study the performance of semiconductor manufacturing processes. Several researchers have discussed the optimal scheduling for a single-cluster tool with residency constraints on transfer and process modules, for example, Rostami et al. (2001) and Rostami and Hamidzadeh (2002). Simulation of cluster tools also plays an important role to study the throughput and optimize the process and design. LeBaron and Hendrickson (2000) and LeBaron and Pool (1994) discussed the use of ASAP to simulate the cluster performance with emulate the real-time cluster tool scheduler. In Dümmler (1999), genetic algorithms are utilized to improve cluster tool performance based on simulation study. Recently, event graph modeling of cluster tools has been studied to accurately capture the throughput and identify bottlenecks for various configurations of the tool (Nehme and Pierce 1994, Pederson and Trout 2002).

Most of above work discussed the single-cluster tool configuration, namely only one robot is serving multiple process and cassette modules. Figure 1(a) shows an example of a single-cluster tool with a single-blade robot. The geometry of the single-cluster tool is compact and analysis and scheduling for such a tool is relative straightforward. For a multi-cluster tool , several single-cluster tools have been

inter-connected through so-called pass-through-modules (or buffer modules) in order to accomplish complicated wafer processes. Figure 1(b) shows an example of a two-cluster tool. For such a multi-cluster tool, wafer flow modeling and scheduling is much more complicated compared with the single cluster tool and analytical methods becomes difficult to solve such problems. It is challenging to simulate such cluster tools since traditional simulation methodologies consider the physical layout of the tool and a multi-cluster tool brings a lot of complexity to the simulation model.

In this paper, we will introduce a highly abstracted methodology to model, simulate and schedule the multi-cluster tools. Event graph modeling approach is employed. We transmute all time consuming tool activities into decision-moving-done cycles that drive the simulation. Instead of detailed layout, event graph based simulation model is very simple and can mathematically manage the activities of multi-cluster tools. Based on such a simulation model, a straightforward tree search method can be applied to find the optimal steady-state scheduling sequence. The fundamental period of cluster tools is calculated based on the minimum cyclic period of all feasible scheduling sequences. Therefore the maximum steady-state throughput can be achieved. A CMP polisher is used as an example of the multi-cluster cluster tools to illustrate the event graph based simulation and scheduling.

The paper is organized as follows. In section 2, we discuss the event graph based simulation modeling for a multi-cluster tool. Section 3 presents the analysis and process scheduling for a multi-cluster tool based on the event graph modeling. An example of the modeling analysis and process scheduling is investigated in section 4 for a CMP polisher. Computational reduction of simulation modeling is also discussed through a model partition method in this section. The concluding remarks are presented in section 5.

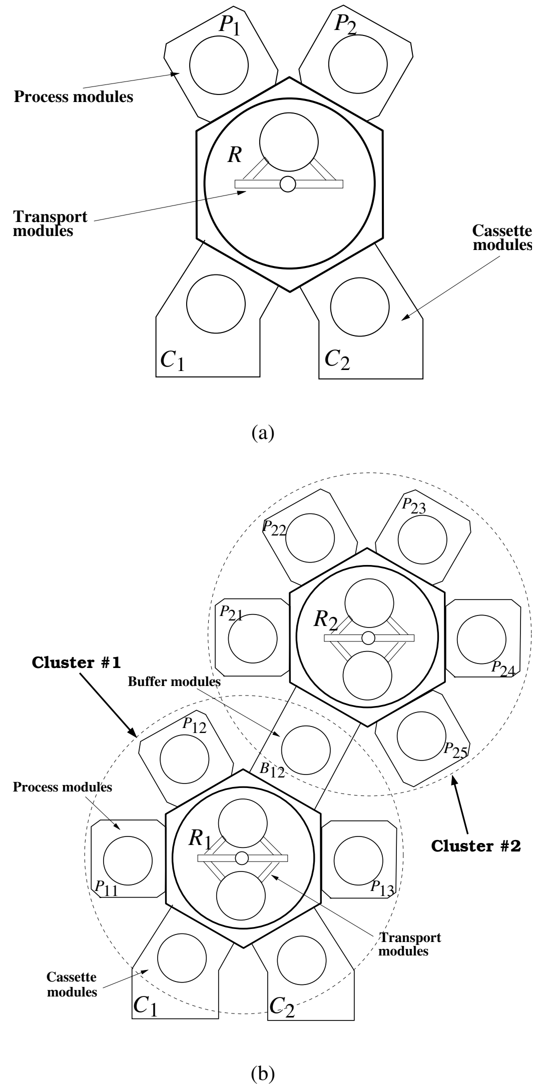

Figure 1: A Schematic of Cluster Tools, (a) Single Cluster Tool, (b) Inter-Connected Multi-Cluster Tool

2 EVENT GRAPH BASED SIMULATION MODEL

2.1 Event Graph

Event graph based modeling and simulation were used for semiconductor manufacturing. Pederson and Trout (2002) discussed the applications of event graph simulation model to the deposition cluster tools. Figure 2 shows an example of the event graph simulation modeling. A discrete event graph model consists of three elements: variables to represent the state of the model, events that will cause the state of the model to change (such as A and B in Figure 2), and the causality relationships between these events ("Statement 1" in Figure 2). Graphically, the events are represented as nodes on a graph and the causality relations are directed edges between the related nodes. Time delays between events and the conditions needed for this edge to be activated are noted on the edges. When an event is scheduled, it can receive arguments called parameters. In the example given by Figure 2, event A will schedule event B to occur t time units in the future if "Statement 1" is true when event A is executed. The scheduled event B will be of type "i". The $\sim$-like shape on the edge indicates that this is a conditional edge. The characteristic of the event graph modeling allows sets of events with similar characteristics to be modeled without having to have duplicate nodes for each cluster (Nehme and Pierce 1994, Pederson and Trout 2002).

In the following sections, we first use an example of a single cluster tool to illustrate the methodology of constructing event graph based simulation model. Then we generalize the methodology for any multi-cluster tools. In this study we assume that there are always wafers in cassette module C_1 that wait for processing and that there are always spaces in cassette C_2 to put wafers in (we use C_i, $i = 1, 2$

to denote cassette modules, and P_i (P_{ji}), $i = 1, \cdots, N$ to denote N process modules within a single cluster (jth cluster).)

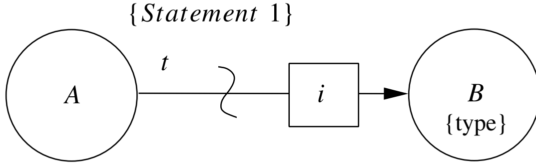

Figure 2: A Schematic of Event Graph Simulation Models

2.2 Event Graph Based Simulation Model: A Single-Cluster Example

We consider an example of a single-cluster tool with two process modules, P_1 and P_2, and a single-blade transfer robot R as shown in Figure 1(a). We consider a single-visit, single route wafer flow without any parallel processing modules. If there exist multiple processing modules that achieve the same process steps, the simulation model discussed in this section should be similar. Denote the wafer flow as: $C_1 \rightarrow P_1 \rightarrow P_2 \rightarrow C_2$, where the arrow stands for wafer transferring by robot R. Figure 3 shows the simulation event graph built for the single-cluster tool in Figure 1(a). In the graph, each solid circle represents one event. For transfer action event, such as "$C_1 \rightarrow P_1$", we represent one node in the graph. For process action events, we represent them by two nodes, one for process starting (e.g. "P_1 proc.") and another for process finishing (e.g. "P_1 done"). We separate the process action events into two nodes because this separation will help us simplify the event graph model in following discussions. Connecting arrows without "$\sim$" sign represents a time-consuming activity. For example, connection from event "$C_1 \rightarrow P_1$" to the event "P_1 proc." represents the activity that the robot takes one wafer from C_1 and put it to P_1. A connection with a $\sim$ represents a conditional trigger of an event (or a decision making). For example, the activity "P_1 done" may trigger an activity of "$P_1 \rightarrow P_2$" if the robot R is available and module P_2 is empty. Detailed conditions, $CND_1, \cdots, CND_7$, are listed in Table 1.

To manage all actions mathematically, we define the state variables vector $\mathbf{S}(t) = [S_i(t)]$ associated with all transfer and process modules. A set of integer values are assigned for each state variable $S_i(t)$, $i = 1, \cdots, N+1$, to represent the all possible state status at time t. Table 2 shows the assigned values of state variables for the example of the single-cluster tool. We also define the action index, denoted as ACT, for all actions for all transfer and process modules. Table 3 shows the values of ACT for all transferring and processing actions discussed above for the example.

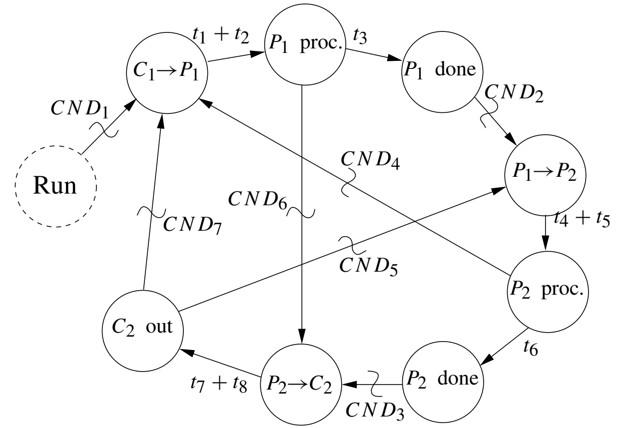

Figure 3: An Event Graph Simulation Modeling for the Single-Cluster Tool Example by Figure 1(a)

In order to handle a more complex transfer actions such as use of double-blade robots, we consider to further separate the transfer activities into two actions: *pick* and *place*. With this separation method, the event graph as shown in Figure 3 can be simplified as decision-moving-done cycles.

Table 1: Causality Conditions for the Example Tool

Label	From	To	Satisfied Conditions
CND_1	Run	$L_1 \rightarrow C_1$	C_1 has wafers, R free P_1 empty
CND_2	P_1 done	$P_1 \rightarrow P_2$	P_1 finishes, R free, P_2 empty
CND_3	P_2 done	$P_2 \rightarrow C_2$	P_2 finishes, R free
CND_4	P_2 proc.	$C_1 \rightarrow P_1$	R free, P_1 empty
CND_5	C_2 out	$P_1 \rightarrow P_2$	R free, P_1 done, P_2 empty
CND_6	P_1 proc.	$P_2 \rightarrow C_2$	R free, P_2 done
CND_7	C_2 out	$C_1 \rightarrow P_1$	R free, P_1 empty

Table 2: State Variable Values $\mathbf{S}$ for the example of single-cluster tool

Values	S_1 (R)	S_2 (P_1)	S_3 (P_2)
-1	In pick/place	In pick/place	In pick/place
0	Free	Free	Free
1	Ready for next action	Ready to process	Ready to process
2		In processing	In processing
3		Process done	Process done

All separated actions in an event graph (Figure 3) can be categorized into three types: *Decision, Move* (or action), and *Done*. The following steps can be employed to simplify the event graph model in Figure 3 into a decision-moving-done cycle shown in Figure 4:

1. All actions shown in Table 3 can be represented by two types of nodes – a starting event "*Move*"

and ending event "*Done*". We can always do so because of action separation.

2. For each action index ACT, define a decision logic **LOG** given by a mapping $f : ACT \rightarrow \mathbf{S}$ as the state values at which action ACT can happen. Table 4 shows the state values when an action can be triggered ("$*$" stands for any arbitrary value). The decision logic can be represented by node "*Decision*" in event graph.
3. At each node "*Move*" for action ACT, we update the state variables $\mathbf{S} \leftarrow \mathbf{S} + \Delta\mathbf{S}(ACT)$, where $\Delta\mathbf{S}(ACT)$ is defined as the state change due to ACT. Table 4 shows the values of $\Delta\mathbf{S}(ACT)$ for each action ACT for the example of the single-cluster tool.
4. At each node "*Done*" for action ACT, we update the state variables $\mathbf{S}$ by a mapping $g(ACT, \mathbf{S}) : \mathbf{S} \times ACT \rightarrow \mathbf{S}$. Table 4 shows the mapping values of $g(\mathbf{S}, ACT)$ for each action ACT for the single cluster example ("I" stands for an identity mapping.)

Table 3: Action Labels for the Example of the Single-Cluster Tool

ACT	**Action**	Nodes in Fig. 3		**Time** t_i (s)
		Move	Done	
1	$C_1 \rightarrow R$	$C_1 \rightarrow P_1$ (pick)	$C_1 \rightarrow P_1$	5
2	$R \rightarrow P_1$	$C_1 \rightarrow P_1$ (plc.)	$C_1 \rightarrow P_1$	5
3	P_1 proc.	P_1 proc.	P_1 done	30
4	$P_1 \rightarrow R$	$P_1 \rightarrow P_2$ (pick)	$P_1 \rightarrow P_2$	5
5	$R \rightarrow P_2$	$p_1 \rightarrow P_2$ (plc.)	$P_1 \rightarrow P_2$	5
6	P_2 proc.	P_2 proc.	P_2 done	35
7	$P_2 \rightarrow R$	$P_2 \rightarrow L_2$ (pick)	$P_2 \rightarrow L_2$	5
8	$R \rightarrow C_2$	$P_2 \rightarrow C_2$ (plc.)	C_2 out	5

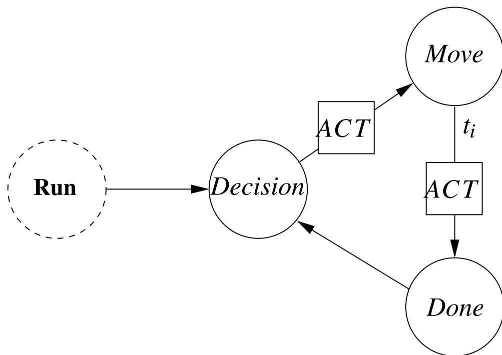

Figure 4: A Simplified Event Graph for Simulation of a Cluster Tool

2.3 Event Graph Based Simulation Model for Multi-Cluster Tools

In the previous section, we described how to construct a event graph simulation model for an example of a single cluster tool. Using a similar method, we can build event graph simulation models for multi-cluster tools as following steps:

1. Define a state variables vector $\mathbf{S}$ that each of its components is associated with each of processing modules, blades of robots and buffer modules between two inter-connected clusters. Table 5 shows some examples of state variable values for different modules.
2. Define an action index variable ACT and assign each action an index value for all transferring and processing actions. Note that each action ACT has been decomposed into two event nodes, "*Move*" and "*Done*".
3. Determine the decision logic mapping $f : ACT \rightarrow \mathbf{S}$ for each action ACT.
4. Construct the state change $\Delta\mathbf{S}(ACT)$ of each action value ACT for the node "*Move*".
5. Construct the state update mapping $g : \mathbf{S} \times ACT \rightarrow \mathbf{S}$ for each action value ACT for the node "*Done*".

The above event graph modeling method can be used to handle random or deterministic timings with various process configurations and any time horizon. Moreover, the decision logic and state update mappings (such as these shown in Table 4) can be automatically generated after we define the state variables, action orders and causality conditions.

Table 4: Decision Logics Mapping f ("*Decision*"), State Change $\Delta\mathbf{S}$ ("*Move*") and State Update Mapping g ("*Done*") for the Example Tool.

ACT		1	2	3	4	5	6	7	8
Decision logic $f : ACT \rightarrow \mathbf{S}$	S_1	0	2	$*$	0	5	$*$	0	8
	S_2	0	0	1	3	$*$	$*$	$*$	$*$
	S_3	$*$	$*$	$*$	0	0	1	3	$*$
$\Delta\mathbf{S}$	S_1	-1	-1	0	-1	-1	0	-1	-1
	S_2	0	-1	$+1$	$+3$	0	0	0	0
	S_3	0	0	0	0	-1	$+1$	-3	0
State update mapping $g : ACT \rightarrow \mathbf{S}$	S_1	2	0	I	5	0	I	8	0
	S_2	I	1	3	0	I	I	I	I
	S_3	I	I	I	I	1	3	0	I

Correctly defining the state variables is an important step in the above creation of the simulation model. We can simply verify the correctness of the state variables values by following the regular process cycle. For example, the state values for a process module follow a cycle of

$0, -1, 1, 2, 3, -1$, and back to 0, which represents the activity sequence: empty, place, start processing, processing, finish processing, pick, and empty again, respectively.

With the simplified event graph model, we can simulate the cluster tools given any scheduling rule. For example, for the priority based scheduling rule (Jevtic 1999), we can simulate the multi-cluster tools in following sequences:

Step 1. Initialize state variable $\mathbf{S}_0$.

Step 2. Match and check the current state variable with the decision logic mapping $f(ACT)$.

Step 3. If no action is executable, go to Step 4. If there is a set of actions that can be executed, say actions $\{ACT_i\}, i = 1, \cdots, m, m \in \mathbb{N}$, we then implement ACT_k, $1 \leq k \leq m$, which has the highest priority number, and update system variable $\mathbf{S}$ by state change mapping $\Delta\mathbf{S}(ACT_k)$ (Table 4). After the status change, we record the finish time of action ACT_k and save it into a stack. Go to Step 2.

Step 4. Check the activities in stack that has started but not finished yet, and move to the nearest finishing time, finish the activity, and update the states $\mathbf{S}$ according to mapping $g(ACT)$ (Table 4).

Step 5. If simulation time reaches, stop simulation; otherwise, go to Step 2.

Table 5: State Variable Values for Multi-Cluster Tools

Modules	Values	Module status
Process	-1	Busy
	0	Ready to place a wafer
	1	With wafers ready to process
	2	In processing
	3	Process done; ready to pickup
One-blade robot	-1	Busy
	0	No wafer on, free for pick
	i	Ready for $ACT = i$
Buffers between clusters i and $i+1$	-1	Busy
	0	No wafer on, ready to place
	1	Wafer ready for cluster i
	2	Wafer ready for cluster $i+1$

The merit of the event graph based simulation model is twofold. First, all activities are abstracted and represented by the same motion cycle: decision, moving, and done. Thus the amount of events is significantly reduced and we can simulate and handle many clusters in a systematic fashion. Second, the whole simulation sequence is accomplished by state checks and state changes on actions. These actions are different only by labels. We can easily transmute the action sequences and check the feasibility of the transmutation. This feature will be further used to search optimal sequences.

3 CLUSTER TOOL SCHEDULING

Scheduling of a cluster tool with identical wafer flows can be decomposed into two phases: in the first phase, a sequence of all robot movings has to be determined; and the second phase is to determine the time for each sequence move. In this section, we follow this two-phase approach. First, we search for all feasible sequences of multi-cluster tool activities. The search algorithm utilizes the event graph based simulation model discussed in previous section. Once we find a set of feasible activity sequences, we determine the optimal sequence and the timing for each move in this sequence, which give a minimal cyclic period (fundamental period). This can be accomplished by either derivative calculation or linear programming (LP) formulation (Rostami and Hamidzadeh 2004). The derivative calculation is employed here for its simplicity. One of advantages of using the event graph based simulation model to determine the optimal scheduling is that the model and sequence searching algorithm are very simple and generic. Thus the algorithm is able to handle some specific requirements such as the process and transfer module residency constraints (Rostami and Hamidzadeh 2004).

For a cluster tool with deterministic processing times, all steady-state activities are done exactly once in a fixed cyclic period, or so-called fundamental period (FP). FP is a relative time period that a wafer enters and leaves visits in a route and those relative times will be repeated for the subsequent wafers in the following periods. Denote FP (in sec.) as the fundamental period for a process on a cluster tool and we can then calculate the cluster tool throughput as $3600/FP$ wafers per hour. It is easy to observe that in order to maximum the throughput performance we need to minimize the FP.

3.1 Searching Feasible Sequences

By applying logic check, state update and change procedures discussed in the previous section, we can easily check all possible activity combinations. The searching algorithm is illustrated by Algorithm 1.

In this algorithm, we try to search all possible activity combinations and calculate the FP at the same time for each feasible activity combination. Since we only want to find activity orders, we can assume zero handling times for all activities. If the tree is fully visited, it is a NP hard problem with computation complexity $O(M!)$, where M is the number of actions. However, since the simulation is implemented inside the search rules, most of the infeasible routes fail the screening of the simulation. As a result, the total amount of searches is significantly decreased. Moreover, a multi-cluster system can be separated into several smaller size sub-systems if some inter-connection configurations apply. Then a large searching problem of the whole

system can be divided into several small problems. Feasibility and complexity of this method will be further discussed in section 3.3.

It is noted that in the event graph model the process actions normally start right after their previous transfer actions finish. Therefore for searching sequences we can always combine the process action events with the robot transfer action events together and reduce the searching action numbers and computing complexity. In section 4, we will show such an example and we can see significantly reduced computations.

Algorithm 1: Search Sequences

input : State information (**S**, **ACT**, decision logics, and causality conditions)
output: All feasible sequence **Seq**
Initialization state $\mathbf{S_0}$
$\mathbf{Seq}_0 \leftarrow \emptyset$; $\mathbf{V} \leftarrow \{\mathbf{ACTs}\}$
$k \leftarrow 0$; $N \leftarrow \dim(\mathbf{V})$
`Search(`$0,\mathbf{Seq}_0,\mathbf{V},\mathbf{S}_0$`)`

function `Search(`$k,\mathbf{Seq},\mathbf{V},\mathbf{S}$`)`
if $k = N$ **then**
  Use Algorithm 2 to calculate **FP**
  Record **Seq** and **FP**
end
for $x \in \mathbf{V}$ **do**
  if x *is feasible* **then**
    $\mathbf{X} \leftarrow \mathbf{V} - x$
    $\mathbf{Y} \leftarrow \{\mathbf{Seq},\ x\}$
    $\mathbf{Z} \leftarrow \mathbf{S}$ with action x move and done
    `Search(`$k+1,\mathbf{X},\mathbf{Y},\mathbf{Z}$`)`
  end
end

One important fact is the choice of the initial state values $\mathbf{S}_0$ in the above searching algorithm. The optimal sequence by Algorithm 1 depends on the choice of $\mathbf{S}_0$. In general, we start the search with following initial state $\mathbf{S}_0$: processes have finished in all process modules and wafers are ready to pickup, and transfer and buffer modules are empty and ready to transfer wafers. If a module is both a buffer and process module (for example, P_{11} of CMP polisher in section 4), then it should be considered as a process module. The proper choice of initial state $\mathbf{S}_0$ can also guarantee the deadlock-free simulation and sequence searching.

3.2 Calculating Scheduling Time for A Sequences

Given a certain robot moving scheduling sequence, we can calculate the FP using Algorithm 2. The algorithm here is to run the same sequence twice with exactly the same order. Then every action x in the given sequence has been running twice and we can calculate the time difference between two xs. The maximum time difference for all actions is the system FP.

The FP calculating algorithm is similar to the simulation procedure discussed in section 2.3. The main difference is that in simulation we must prioritize actions to facilitate the decision when two or more activities conflict. The algorithm instead utilizes a determined feasible sequence and calculates the FP.

Algorithm 2: FP Calculation for a Feasible Sequence

input : A feasible sequence $Seq \in \mathbf{Seq}$
output: Fundamental period **FP** for the sequence *Seq*
Initialization state $\mathbf{S_0}$
RepReq $\leftarrow \{Seq + Seq\}$; $\mathbf{S}_a \leftarrow \emptyset$
for $x \in$ **RepSeq do**
  while x *not allowed to move with state* $\mathbf{S}_t$ **do**
    Find the first finishing process x_f in $\mathbf{S}_a$
    POP x_f
    Update state $\mathbf{S}_t$ as x_f *Done* (mapping g)
    $t \leftarrow \max(t(x_f), t)$
  end
  Update state $\mathbf{S}_t$ as x *Move* (by $\Delta\mathbf{S}$)
  $t_f \leftarrow t + t(x_f)$
  PUSH $\{x,\ t_f\}$ into stack $\mathbf{S}_a$
end
FP $= \max_{x \in \mathbf{Req}}$ (Time diff. between two x actions in **RepSeq**)

Special process requirements affect the scheduling of cluster tool. In general, these requirements could be considered and implemented in the algorithm to find an optimal schedule. For example, residency constraints are found in processes such as chemical vapor deposition (CVD) and rapid thermal processing, where leaving a wafer in a process module longer than its processing time (i.e., over-processing) can be detrimental to wafer desired qualities (Rostami et al. 2001). To incorporate such requirements into algorithms, we can modify the above algorithm and enforce a set of maximum delay time constraints for each action. By doing so, we still obtain the same set of feasible sequences without considering these constraints by Algorithm 1 but we only search optimal scheduling with residency constraints through Algorithm 2.

3.3 Partition of Multi-Cluster Tools

The throughput of multi-cluster tools represents the throughput of each individual sub-systems. Otherwise some modules will be blocked or starved by wafer flows. Therefore, if we could decompose a multi-cluster tool into several smaller size single-cluster tools that operate as "independently", then the maximum of FPs of these single clusters

is the FP that the whole integrated system can bear. Based on this observation, the partition of the multi-cluster tools could be effective way to reduce the complexity of the scheduling search.

It is natural to consider a single-cluster (robot) partition in which each single-cluster is considered as a sub-system. However, due to the complex configurations of the buffer modules and transfer modules between two inter-connected clusters, we cannot simply decouple two clusters as they were separated and independent. Here we only consider a special case when two inter-connected clusters are both equipped with double-blade robots and the capacity of the buffer modules between them is equal or more than two wafers (for more detail discussion of a general decomposition approach for multi-cluster tools readers can refer to Yi et al. (2004).) Clusters #1 and #2 in Figure 6 are such an example. Using this example, we explain some decomposition ideas briefly. The buffer modules, B_{11} and B_{12}, are also serving as process modules P_{11} and P_{12}. Figure 5 shows the Gannt chart of a scheduled buffer processing modules B_{11} (P_{11}) and B_{12} (P_{12}) that can be used to decompose these two clusters. T_1 is the time for R_1 placing a wafer into B_{11}, T_2 is the time for $B_{11}(P_{11})$ processing, T_3 is the time for R_2 picking up a wafer from B_{11}, T_4 is the time for R_1 picking up a wafer from B_{12}, T_5 is the time for R_2 placing a wafer into B_{12}, and T_6 is the time for $B_{12}(P_{12})$ processing. In order to decouple these two clusters, we can arrange the following timing and scheduling requirements:

1. For each buffer, no-overlap (time conflicts) between processing and wafer transferring.
2. Timings for "pick" from one buffer and "place" to another buffer are considered in the FP calculations of both clusters.

Once the two clusters' schedules satisfy requirement 2, the relative time between T_1 and T_4, and between T_3 and T_5 are fixed. By changing the time difference between T_1 and T_3 (T_4 and T_5 synchronously) by requirement 1, the scheduling will work for both clusters. Such a partition approach can be easily taken within the decision-moving-done simulation context. Thus a problem of scheduling of a 4-cluster system can be separated into two scheduling problems of a 3-cluster and a single-cluster sub-systems respectively. Computational results for this example will be discussed in the next section. In general, if a problem of size M can be decomposed into L sub-problems, and each of them has smaller size M_i, where $\sum_i^L M_i = M$, then computation complexity becomes $\sum_i^L O(M_i!)$ (instead of $O(M!)$ before.) This partition technique significantly reduces searching time.

4 EXPERIMENT EXAMPLES

Figure 6 shows a schematic of a CMP polisher used in semiconductor manufacturing industry. Chemical-mechanical planarization process is widely used in industry to planarize the wafer surface and therefore enhance the photolithograph process performance. The CMP polisher can be modeled as a four-cluster tool. There are two double-blade robots R_1 and R_2, an single-blade robot R_3 and an indexer R_4. The indexer R_4 moves wafers simultaneously from processing modules P_{41} to P_{42}, P_{42} to P_{43}, P_{43} to P_{44} and P_{44} to P_{41}, respectively. The wafers go through the cluster tool as following flow chart (we use the notation R_{ij} for the blade j of the robot i):

$$C_1 \xrightarrow{R_{11}} P_{11}(B_{11}) \xrightarrow{R_{21}} B_{21} \xrightarrow{R_3} P_{41} \xrightarrow{R_4} P_{42} \xrightarrow{R_4} P_{43} \xrightarrow{R_4} P_{44} \xrightarrow{R_4} P_{41} \xrightarrow{R_3} B_{21} \xrightarrow{R_{22}} P_{21} \xrightarrow{R_{22}} P_{22} \xrightarrow{R_{22}} P_{12}(B_{12}) \xrightarrow{R_{12}} C_2$$

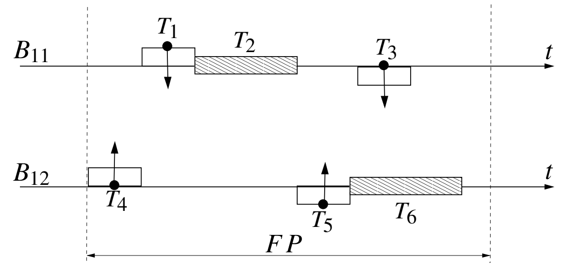

Figure 5: Gannt Chart of Buffer Modules B_{11} and B_{12}

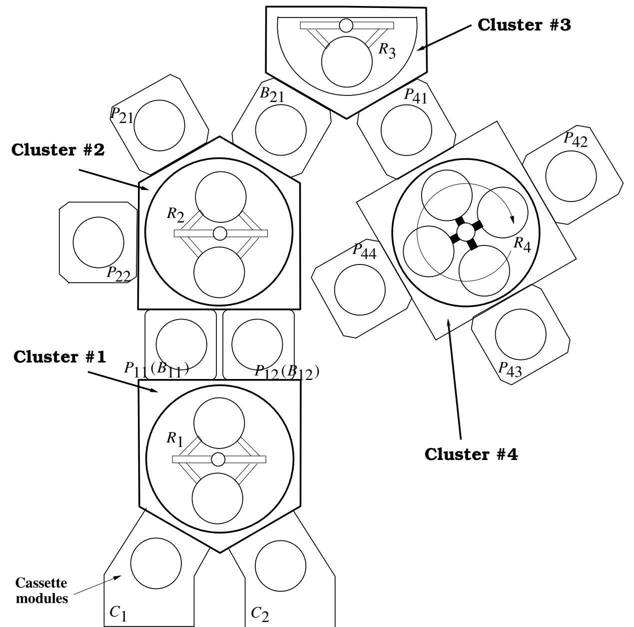

Figure 6: A Simplified CMP Multi-Cluster Polisher

Simulation modeling of the CMP tool can be handled as follows. First, we define the state variables and action index variable. A decision-move-done event graph can be constructed for the whole cluster tool. There are total 15 states (total number of process and transfer modules) and 20 actions (as shown in Table 6) for the event graph simulation model. The actions are selected such that (1) each process has one identity, (2) each single-blade robot action has one identity (pick and place tied together), and (3) each double-blade robot has two identities (pick and place separated) because they may be taken in different sequences. The states are selected in the way that each possible buffer unit is represented. Robot R_3 and indexer R_4 cannot hold wafers independently, and thus are not considered as independent decision states. The decision logic, state update and change mappings can be built accordingly. The initial states are set such that every processing module has a finished wafer ready for pickup and other modules are idle. We create such a event graph modeling and simulation using simulation package SIGMA (Schruben 2000).

We utilize the simulation model and algorithms in section 3 to find the optimal schedules. The experiments are carried out on a computer with Centrino 1.3GHz CPU and 256M RAM and the computation results are listed in Table 7. We first search all actions for any possible visits and it takes about 3 hours computing time. After binding process actions with robot movements as we mentioned in section 3.1, the new model finds only 1508 possible schedules within 4.1 secs. From this example, it is clear that combining process actions with their previous transfer actions significantly reduces the computation load (by more than 1000 times.) The resulting FP is 136 secs./wafer, namely the steady-state throughput for the cluster tool is 26.5 wafers per hour. The real throughput of such a CMP polisher is 26 wafers per hour using a priority rule based scheduler.

Due to the similarity among clusters R_1 and R_2, we can apply the partition approach discussed in section 3.3 to decouple the whole systems into two sub-systems: one only including Cluster #1 and the other sub-system consisting of Clusters #2-4. From the computing results given by Table 7, the partition method significantly reduces the complexity (around 20 times faster.)

In practice, the number of actions taken by any single cluster is limited and thus the computation is not heavy. The proposed algorithms can solve most practical multi-cluster tool simulation and scheduling problems efficiently. For the example of CMP polisher, cluster R_2 is the bottleneck component in the whole cluster tool. After denoting the bottleneck of the tool, it is easy to find the effect of system change on the throughput. When one process takes longer time as expected, it won't change the system throughput if it is not within the bottleneck cluster and the change is not large enough to become the new bottleneck. In fact, we find that the optimal schedules of CMP polisher are not unique (Table 7.) This observation implies that the schedule of the whole multi-cluster tool might not be unique since the schedules of non-bottleneck clusters can be varying if the resulting FP is smaller than the FP of the bottleneck cluster. This characteristic of multi-cluster tools could result in the fact that a priority based schedule rule could produce a maximum throughput if we set the bottleneck cluster actions as the highest priorities.

Table 6: Activities of the CMP Polisher

ACT	**Action**	**Time** t_i (s)
0	$C_1 \xrightarrow{R_{11}} P_{11}(B_{11})$	15
1	P_{11} processing	10
2	$P_{11}(B_{11}) \xrightarrow{R_{21}} B_{21}$ first half	10
3	$P_{11}(B_{11}) \xrightarrow{R_{21}} B_{21}$ second half	17
4	$B_{21} \xrightarrow{R_3} P_{41}$	18
5	$P_{41} \xrightarrow{R_4} P_{42}$, $P_{42} \xrightarrow{R_4} P_{43}$, $P_{43} \xrightarrow{R_4} P_{44}$, $P_{44} \xrightarrow{R_4} P_{41}$	5.5
6	P_{42} processing	60
7	P_{43} processing	60
8	P_{44} processing	60
9	$P_{41} \xrightarrow{R_3} B_{21}$	26
10	$B_{21} \xrightarrow{R_{21}} P_{21}$ first half	18
11	$B_{21} \xrightarrow{R_{21}} P_{21}$ second half	20
12	P_{21} processing	30
13	$P_{21} \xrightarrow{R_{21}} P_{22}$ first half	15
14	$P_{21} \xrightarrow{R_{21}} P_{22}$ second half	20
15	P_{22} processing	30
16	$P_{22} \xrightarrow{R_{22}} P_{12}(B_{12})$ first half	16
17	$P_{22} \xrightarrow{R_{22}} P_{12}(B_{12})$ second half	20
18	P_{12} processing	20
19	$P_{12}(B_{12}) \xrightarrow{R_{12}} C_2$	20

5 CONCLUSIONS

In this paper, we presented an event graph based simulation and scheduling analysis of a multi-cluster tool for semiconductor manufacturing processes. A decision-moving-done event graph model is utilized to study the scheduling problem of multi-cluster tools. The proposed event graph method facilitates the optimal scheduling search algorithm and throughput calculations. By emulating the simulation into the search algorithm and combining some process actions with transfer actions, the simulation modeling significantly cut the search tree branches. Thus the computation time and complexity is manageable. Partition of a com-

Table 7: Testing Results of the CMP Polisher with a Real Throughput 26 Wafers per hour by a Priority Based Scheduler

Clusters	Actions	Total acts.	Search routes	Comput. time (s)	Feasible scheds.	FP (s)	Optimal scheds.
All	All actions	20	75489480	10428	1770120	136	300960
All	Proc. combined	13	50022	8.7	4428	136	296
R_1,R_2 partition	Proc. combined	11	3313	0.4	175	136	19

plex multi-cluster tool into several smaller size sub-systems can be employed to reduce the complexity further under a certain buffer module configurations. An application example of chemical-mechanical planarization (CMP) polisher illustrates the efficiency and complexity of the proposed simulation and scheduling method.

REFERENCES

Dümmler, M. A. 1999. Using Simulation and Genetic Algorithems to Improve Cluster Tool Performance. In *Proceedings of the 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard, 875–879. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Jevtic, D. Method and aparatus for priority based scheduling of wafer processing within a multiple chamber semiconductor wafer processing tool. U.S. Patent 5,928,389, July, 1999.

LeBaron, H. T., and R. A. Hendrickson. 2000. Using Emulation to Validate a Cluster Tool Simulation Model. In *Proceedings of the 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. R. Barton, 1417–1422. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

LeBaron, H. T., and M. Pool. 1994. The Simulation of Cluster Tools: A New Semiconductor Manufacturing Technology. In *Proceedings of the 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan, 907–912. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Nehme, D. A., and N. G. Pierce. 1994. Evaluation the Throughput of Cluster Tools Using Event-graph Simulations. In *Proceedings of the 1994 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 189–192. Cambridge, MA.

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

Perkinson, T., P. McLarty, R. Gyurcsik, and R. Cavin. 1994. Single-Wafer Cluster Tool Performance: An Analysis of Throughput. *IEEE Trans. on Semiconductor Manufacturing* 7 (3): 369–373.

Rostami, S., and B. Hamidzadeh. 2002. Optimal Scheduling Techniques for Cluster Tools With Process-Module and Transport-Module Residency Constraints. *IEEE Trans. on Semiconductor Manufacturing* 15 (3): 341–349.

Rostami, S., and B. Hamidzadeh. 2004. An Optimal Residency-Aware Scheduling Technique for Cluster Tools With Buffer Module. *IEEE Trans. on Semiconductor Manufacturing* 17 (1): 68–73.

Rostami, S., B. Hamidzadeh, and D. Camporese. 2001. An Optimal Periodic Scheduler for Dual-Arm Robots in Cluster Tools with Residency Constraints. *IEEE Trans. Robot. Automat.* 17 (5): 609–618.

Schruben, L. W. 2000. *SIGMA User's Guide*. Berkeley, CA: Department of Industrial Engineering and Operations Research, University of California at Berkeley.

Srinivasan, R. 1998. Modeling and Performance Analysis of Cluster Tools Using Petri Nets. *IEEE Trans. on Semiconductor Manufacturing* 11 (3): 394–403.

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 Trans. on Semiconductor Manufacturing* 10 (4): 418–424.

Yi, J., D. Song, and S. Ding. 2004. Steady-State Throughput and Scheduling Analysis of Multi-Cluster Tools for Semiconductor Manufacturing Using Decomposition Methods. Working Paper.

Zuberek, W. 2001. Timed Petri Nets in Modeling and Analysis of Cluster Tools. *IEEE Trans. on Robotics and Automation* 17 (5): 562–575.

AUTHOR BIOGRAPHIES

SHENGWEI DING received his B.S. and M.S. in electrical engineering from Zhejiang University, China, in 1996 and 1999, respectively. He received the M.S. and Ph.D. degree in industrial engineering and operation research from the University of California at Berkeley in 2000 and 2004, respectively. He is a member of IIE and INFORMS. His research interests are simulation, scheduling, queuing models and production management in semiconductor manufacturing. His e-mail address is <dingsw@ieor.berkeley.edu>.

JINGANG YI is a systems engineer in Lam Research Corporation. He received his B.S. in electrical engineering from Zhejiang University, M.Eng. in precision instruments from Tsinghua University, M.A. in applied mathematics and Ph.D. in mechanical engineering from the University of California

at Berkeley in 1993, 1996, 2001 and 2002, respectively. His research interests include intelligent control and automation and its applications to semiconductor manufacturing. Dr. Yi is a member of IEEE and ASME. His e-mail address is <jingang.yi@cal.berkeley.edu>.