A Scheduling Approach Using RSEM for Dedicated Machine Constraint

Abstract

The dedicated machine constraint for photolithography process in semiconductor manufacturing is one of the new challenges introduced in photolithography machinery due to natural bias. With this constraint, the wafers passing through each photolithography process have to be processed on the same machine. The purpose of the limitation is to prevent the impact of natural bias. However, many scheduling polices or modeling methods proposed by previous research for the photolithography machines in the semiconductor manufacturing system have not discussed the dedicated machine constraint. We propose an interactive scheduling method, the Load Balancing (LB) approach using the Resource Schedule and Execution Matrix (RSEM) as a tool to tackle this constraint. The LB approach is to schedule each wafer lot at the first photolithography stage to a suitable machine according to the load balancing factors among machines. We describe the algorithm of the proposed LB approach and RSEM in the paper. We also present an example to demonstrate our approach and the result of the simulations to validate our approach.

Introduction

One of the challenges in the semiconductor manufacturing systems is the dedicated photolithography machine constraint which is caused by the natural bias of the photolithography machine. Natural bias will impact the alignment of patterns between different layers. The smaller the dimension of the IC products (wafers), the more difficult they will be to align between different layers. A study discussing the performance improvements for photolithography process terms the dedicated constraint as machine dedication policies (Akcalt, Nemoto, and Uzsoy 2001). This is the case especially when we move on to a smaller dimension IC for high technology products. The wafer lots passing through each photolithography stage have to be processed on the same machine. The purpose of the limitation is to prevent the impact of natural bias and to keep a good yield of the IC product. Figure 1 describes the dedicated machine constraint. When wafer lots enter each photolithography operation stage, with this constraint, the wafer lots dedicated to machine ***X***, they need to wait for machine ***X***, even if there is no wafer lot waiting for machine ***Y***, which is idle. On the other hand, when wafer lots enter into other operation stages, without any machine constraints, the wafer lots can be scheduled to any machine of A, B or C as long as it becomes idle.

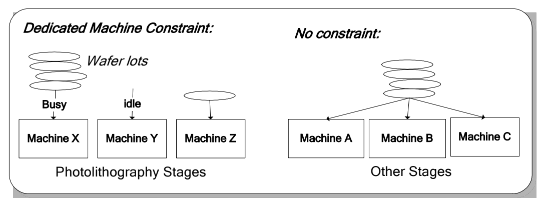


Figure 1: Dedicated photolithography machine constraint

Semiconductor manufacturing systems are different from the traditional manufacturing systems. In a semiconductor factory, one wafer lot passes through hundreds of operations, and the processing procedure takes a few months to complete. The operations of semiconductor manufacturing incrementally develop an IC product layer by layer. A “Re-Entrant” production line has been proposed to model the process flow of a semiconductor manufacturing system (Kumar 1993) (Kumar 1994).

The scheduling problems of the photolithography machines have been studied by some researchers. Their proposed scheduling methods make an effort to improve the performance of the photolithography machines. Arisha and Young proposed a Neural Network approach to improve the performance of the photolithography machines (Arisha and Young 2004). Mönch, Prause, and Schmalfuss worked on the wafer lots assignment of the photolithography machines to improve the load balancing problem (Mönch, Prause, and Schmalfuss 2001). The scheduling rule, Stepper Dispatch Algorithm (SDA-F) (Chern and Liu 2003), uses a rule-based algorithm to dispatch wafer lots in photolithography stages following least slack principles. However, their proposed scheduling methods did not concern the dedicated machine constraint. The constraint is the most important challenge to improve productivity and fulfill the request for customers as well as the main contributor to the complexity and uncertainty of semiconductor manufacturing. If we randomly schedule the wafer lots to arbitrary photolithography machines at the first photolithography stage, then the load of all photolithography machines might become unbalanced. This load balancing issue derived mainly from the dedicated photolithography machine constraint. This happens because once the wafer lots have been scheduled to one of the machines at the first photolithography stage, they must be assigned to the same machine in the subsequent photolithography stages until they have passed the last photolithography stage. Therefore, the short time of unexpected breakdown of one machine will cause a pile-up of many wafer lots waiting for the machine and the situation makes the machine critical to the factory.

Therefore, the unbalanced load among photolithography machines will mean that some of the photolithography machines will become idle and remain so for a while, due to the fact that no wafer lots can be processed, and the other will always be busy while many wafer lots in the buffer limited to this machine are awaiting processing. As a result, the performance of the factory will have been decreased and impacted. The wafer lots of a load unbalancing factory usually need to be switched from the highly congested machines to the idle machines. It relies on experienced engineers to manually handle alignment problem of the wafer lots with a different situation off-line. It is inefficient to determine one lot at a time which wafer lot and machine need be switched. Moreover, this method cannot meet the fast-changing market of the semiconductor industry.

Motivated by the issues described above, we propose an interactive scheduling method applying the Load Balancing (LB) approach and the Resource Schedule and Execution Matrix (RSEM) to undertake the dedicated machine constraint (Shr, Liu, and Chen 2006a) (Shr, Liu, and Chen 2006b). The LB approach is to schedule each wafer lot for a photolithography machine at the first and unconstrained stage. The assignment is according to the load factor among photolithography machines generated by the RSEM in real time. For all other stages, the LB approach follows the schedule rule which is similar to the policy used by Least Slack time methods. We can also introduce new or cancel orders within the LB scheduling process.

The rest of the paper is organized as follows. First, we describe the RSEM and proposed LB approach, and then we will use an example to demonstrate the LB approach. After that, we will state the simulation result that validated the LB approach. We conclude with the discussions and our intended future work in the end.

Resource Schedule and Execution Matrix

We introduce the Resource Schedule and Execution Matrix (RSEM) by describing its algorithm and architecture in detail. The RSEM method consists of three modules *Task Generation*, *Resource Calculation*, and *Resource Allocation* modules.

Task Generation. The first module is to model the tasks for the scheduling system. For example, in the semiconductor factory, the tasks are the procedures of processing wafer lots, starting from the raw material until the completion of the IC products. We generate a two-dimension matrix for the tasks that are going be processed by machines. One dimension is reserved for the tasks $t_1, t_2, \ldots, t_n$, the other is to represent the periodical time event (or step) $s_1, s_2, \ldots, s_m$. Each task has a sequential pattern to represent the resources it needed during the process sequence from a raw material to a product. We define each type resource as $r_1, r_2, \ldots, r_o$, where it means a particular task needs the resources in the sequence of r_1 and r_2 following that until r_o is gained. Therefore, the matrix looks as follows:

	s_1	s_2	.	.	.	.	.	s_i	.	.	s_m
t_1	r_1	r_2	r_3	..	..	..	r_k	..	..	..	..
t_2		r_3	r_4	..	..	r_k	..	..	..	..	..
.							..	..			
t_i					r_3	r_4	..	..	r_k	..	
.							.	.			
t_n					..	..	r_k	..	..	..	

The symbol, r_k in the Matrix[t_i, s_j] is to represent the fact that the task t_i needs of the resource (machine) r_k at the time s_j. If t_i starts to be processed at s_j and the total step numbers of t_i is p, we will fill its pattern into the matrix from Matrix[t_i, s_j] to [t_i, s_{j+p-1}]. All the tasks, $t_1, \ldots t_n$, follow the illustration above to form a task matrix in the task generation module. To represent the dedicated machine constraint in the matrix for this research, the symbol r_k^x, a replacement of r_k, is to represent that t_i has been dedicated to number x of type k machine at s_j. The symbol w_k is to represent the wait situation when the machine r_k cannot serve t_i at s_j. We will insert this symbol in the *Resource Allocation* module later.

Resource Calculation. The *Resource Calculation* module is to summarize the value of each dimension as the factors for the scheduling rules of the *Resource Allocation* module. For example, we can determine how many steps t_i needed to be processed by counting task pattern of t_i dimension in the matrix. We can also realize how many wait steps t_i has had by counting w_k from start step to current step of t_i dimension in the matrix. Furthermore, if we count the r_k^x in s_j dimension, we can know how many tasks will need the machine m_x of r_k at s_j. Some definitions and formulae of these factors in the *Resource Calculation* module are as follows:

Required resource (machine):

How many tasks will need the resource r_k at s_j?

$$RR(r_k, s_j) = \sum Matrix[t_i, s_j] = r_k \,. \tag{1}$$

How many tasks will wait for the resource r_k at s_j?

$$QueueBuf(r_k, s_j) = \sum Matrix[t_i, s_j] = w_k \tag{2}$$

Count steps of tasks:

How many wait steps t_i has had before current step?

$$WaitStep\ (t_i) = \sum_{j=start}^{current\ \ step} Matrix\,[t_i, s_j] = w_k \,, 1 \le k \le o \tag{3}$$

How many steps t_i will have?

$$Steps\,(t_i) = \sum_{j=start}^{end\ step} Matrix\,[t_i, s_j] \neq Null \qquad \textbf{(4)}$$

Resource Allocation. Before we can start the execution of the *Resource Allocation* module, we need to generate the task matrix, obtain all the factors for the scheduling rules, and build up the rules. The module is to schedule the tasks to the suitable resource according to the factors and predefined rules. To represent the situation of waiting for r_k; i.e., when t_i can not take the resource of r_k at the time s_j, then we will not only insert w_k in the pattern of t_i, but also need to shift the following pattern to the next step in the matrix. Therefore, we can obtain the updated factor for how many tasks wait for r_k at s_j only if we have counted w_k by the dimension s_j (formula (2)). We can also obtain the factor for how many wait step that t_i has had only if we have counted w_k, $1 \le k \le o$ by t_i dimension in the matrix.

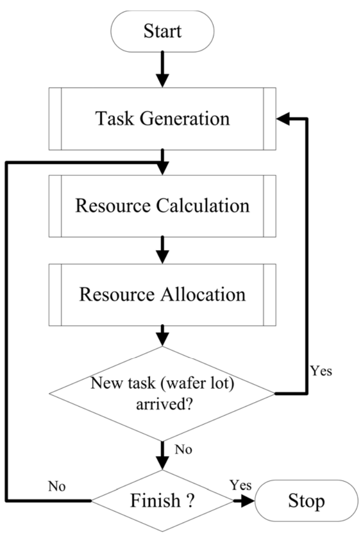


Figure 2: Flowchart of the RSEM

Flowchart and Algorithm. To better *understand* our proposed scheduling process, the flowchart of RSEM is shown in Figure 2. The process of using the RSEM starts from the *Task Generation* module and it will copy the predefined task patterns of tasks into the matrix. Entering the *Resource Calculation* module, the factors for the tasks and resources will be brought out at the current step. This module will update these factors again at each scheduling step. The execution of scheduling process is in the *Resource Allocation* module. When we have done the schedule for all the tasks for the current step, we will return to check for new tasks and repeat the whole process again by following the flowchart. We will exit the scheduling process when we reach the final step of the last task if there is still no new task appended to the matrix. After that, the scheduling process will restart immediately when the new tasks arriving in the system.

To be more concrete, three algorithms for the *Task Generation* (**Algorithm 1**), *Resource Calculation* (**Algorithm 2**), and *Resource Allocation* (**Algorithm 3**) modules are depicted as follows.

In **Algorithm 1**, the procedure appends tasks to the system by copying the task patterns of the tasks in the matrix. It will start from the start step s_s to the end step s_e of each task. The s_s will not start before the current step s_c as well as that the s_e should not end beyond the maximum step s_m of the matrix in the system. The task matrix will be passed to and manipulated at the following algorithms. We will have some factors ready for scheduling after the *Resource Calculation* in **Algorithm 2**. For example, there are four factors, **Total_Step(**t_i**)** and **Wait_Step(**t_i**)** for the tasks, and **Resource_Demand(**r_k**)** and **Queue_Buffer(**r_k**)** for the resources. We obtain these factors by simply counting the occurrences of the desired symbols like r_k or w_k, along the specific task t_i dimension or the current step s_c of the task matrix. We also can include other factors in this module depending on different applications, e.g., the factors of the load of photolithography machine and the remaining photolithography stages of the tasks in the example of next Section.

The procedure of **Algorithm 3** is to execute the scheduling process for the tasks and resources. The first part of the scheduling process is to allocate all the available resources to optimize the performance or production goals of the manufacturing system, but it must satisfy all the constraints. The scheduling rule of the proposed Load Balancing approach is one of the examples. After the process for resources allocation, the second part of the scheduling process is to insert a wait step and shift a step for all the tasks which are not assigned for a machine. A wait symbol w_k is waiting for machine type k and a w_k^x is waiting for dedicated machine number x, m_x of machine type k.

Algorithm 1 ***Task_Generation*** {
// Copy task patterns of t_i into matrix
// from start step, s_s, to end step, s_e,
// $s_c \le s_s \le s_e \le s_m$, s_m: max step in system.
// s_c: current step, l: length of task patterns for t_i
for $i = 1$ *to* n
    for $j = s_s$ *to* s_e; $k = 1$ *to* l
        matrix[t_i, s_j] = Taskpatterns[t_i, r_k]
    next
next
}

Algorithm 2 ***Resource_Calculation*** {
//Factor for tasks, function: **Total_Step(**t_i**)**
//To count total steps of tasks
//n: total tasks, m: max step in system.
for $i = 1$ *to* n
    for $j = 1$ *to* m
        if (matrix[t_i, s_j] ≠ ***null***) ***then***
            Total_Step(t_i**)**= +1; /*Count total steps*/
    next
next
//Factor for tasks, function: **Wait_Step(**t_i**)**
//To count total wait steps of tasks, s_c: current step.
for $i = 1$ *to* n
    for $j = 1$ *to* s_c
        if (matrix[t_i, s_j] = w_k) ***then***

Wait_Step(t_i)= +1; /*Count wait steps*/
next
next
//Factor for Resource, function: **Resource_Demand(r_k)**
//To count total tasks which are need of the resource r_k
// $k = 1,\ldots,o$, o: total resource.
for $k = 1$ *to* o
for $i = 1$ *to* n
if (matrix[t_i, s_c] = r_k) ***then***
Resource_Demand(r_k)=+1;
next
next
//Factor for Resource, function: **Queue_Buffer(r_k)**
//To count tasks which are waiting for of the resource r_k
for $k = 1$ *to* o
for $i = 1$ *to* n
if (matrix[t_i, s_c] = w_k) ***then***
Queue_Buffer(r_k)=+1;
next
next
:……..
}

Algorithm 3 ***Resource_Allocation*** {
//Scheduling; o: total resource.
for $k = 1$ *to* o
Assign tasks to r_k, according to predefined rules
e.g., the load balancing scheduling rules in the example of next Section.
next
//Execution; shift task pattern of the tasks,
//which does not be scheduled at current step;
//x: the order number of resources.
for $i = 1$ *to* n
if (t_i dose not allocate the resource) ***then***
insert w_k to wait for r_k;/**without dedicated constraint**/
or
insert w_k^x to wait for m_x of r_k;/**with dedicated constraint* */
next
}

We use the RSEM to represent complex tasks and allocate resources by the simple matrix calculation. This reduces much of the computation time for the complex problem. The RSEM provides two kinds of functions. One is that we can follow the predefined rules from expert knowledge to obtain the resource allocation result at each step quickly by the factors summarized from task matrix. The other is that we could predict the bottleneck or critical situation quickly by executing proper steps forward. This can also evaluate the predefined rules to obtain better scheduling rules for the system at the same time.

Load Balancing Scheduling Approach

In this section, we apply the Load Balancing (LB) approach to the dedicated photolithography machine constraint in semiconductor manufacturing. Our proposed LB method uses the RSEM as a tool to represent the temporal relationship between the wafer lots and machines during each scheduling step. First, RSEM spans all the water lots as a two-dimension matrix to represent the activities of tasks in the factory. Second, the LB method is to obtain some value of factors from RSEM. The factors are the summary of count or sum of the rows or columns by RSEM. The final step is to execute the scheduling. Our proposed LB scheduling approach will assign each wafer lot at the first and unconstrained photolithography stage to a suitable photolithography machine. The rule for the assignment is to select a wafer lot with biggest wait steps to a smallest load photolithography machine. These wafer lots will be put into the queue buffer of their assigned photolithography machine. As soon as these wafer lots have been assigned, the LB approach will start to schedule a wafer lot for each photolithography machine. The method for the scheduling is to select the biggest wait step in the queue buffer of each photolithography machine to this photolithography machine. This method is similar to the scheduling method used by the Least Slack time approach. We can also introduce new or cancel orders within the scheduling process. We use an example to demonstrate the approach.

After obtaining the process flow for customer product from the database of semiconductor manufacturing, we can use a simple program to transform the process flow into our matrix representation. For a typical factory, there exist thousands of wafer lots and hundreds of process steps. We start from transforming the process pattern of wafer lots into a task matrix. We let r_2 represent the photolithography machine and r_k, $k \neq 2$, to represent non-photolithography machines. We use different numbers of r_k together, e.g., r_k, $r_k r_k$, or $r_k r_k r_k r_k, \ldots$, for the task patterns to represent different process time of different photolithography and non-photolithography stages. The symbol r_2^x in the Matrix[i,j] is to represent the wafer lot t_i need of the photolithography machine m_x at s_j with dedicated machine constraint, while r_k^x $(k \neq 2)$ is to represent the t_i need of the machine type k and the machine m_x at $s_{j'}$ $(j \neq j')$ without dedicated machine constraint. There is no assigned machine number for the photolithography machine before the wafer lot has passed first photolithography stage. Suppose that the required resource pattern of t_1 is as follows:

$$r_1 r_3 r_2 r_4 r_5 r_6 r_7 r_2 r_2 r_4 r_5 r_6 r_7 r_7 r_9 r_1 r_3 r_2 r_2 r_2 r_2 r_4 r_5 r_6 r_7 r_3 r_2 r_2 r_2 r_6 r_6 r_7 r_5 r_5 r_9$$

and starts the process in the factory at s_1. We will fill its pattern into the matrix from Matrix[t_1,s_1] to Matrix[t_1,s_{35}], which indicates that the total number of the steps for t_1 is 35. The following matrix shows the pattern of this wafer lot t_1. Two examples are used to demonstrate two different situations, with or without dedicated machine constraint, which will be depicted in the matrix. To consider the dedicated machine constraint in the first example is as follow: one wafer lot t_2 in the matrix having the same required resource pattern as t_1 but starting at s_3. The wafer lot t_i in the matrix starts from s_8, and then it requires the same type resource, the photolithography machine, but does not have the same (number) machine at the step s_{11}. This represents that t_2 needs the machine m_1, while t_i has not been dedicated to any machine yet. An example without dedicated machine constraint is as follows: at s_{12}, two tasks, t_2 and t_i, might compete for the same resource r_4 if the resource of r_4 is not enough for them at that time s_{12}.

s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} s_{16} s_{17} s_{18} s_{19} s_{20} s_{21} s_{22} s_{23} s_{24} s_{25} s_{26} s_{27} s_{28} s_{29} s_{30} s_{31} s_{32} s_{33} s_{34} s_{35} .. .. .. s_i .. .. s_m

t_1 r_1 r_3 r_2 r_4 r_5 r_6 r_7 r_2 r_2 r_4 r_5 r_6 r_7 r_7 r_9 r_1 r_3 r_2 r_2 r_2 r_2 r_4 r_5 r_6 r_7 r_3 r_2 r_2 r_2 r_6 r_6 r_7 r_5 r_5 r_9 .

t_2 r_1 r_3 r_2 r_4 r_5 r_6 r_7 $\mathbf{r_2}^{1}$ $\boxed{\mathbf{r_2}^{1}}$ $\boxed{\mathbf{r_4}}$ r_5 r_6 r_7 r_7 r_9 r_1 r_3 r_2 r_2 r_2 r_2 r_4 r_5 r_6 r_7 r_3 r_2 r_2 r_2 r_6 r_6 r_7 r_5 r_5 r_9

t_i .. r_1 r_3 $\boxed{\mathbf{r_2}}$ $\boxed{\mathbf{r_4}}$ r_6 r_5 .. ..

.. .. ..

The formula (5) is derived from formula (1) to define the dedicated machine constraint in the system (P is the total numbers of resource r_k):

$$RR(r_k^x, s_j) = \sum_{t_i \in Wafer\ lots} Matrix[t_i, s_j] = r_k^x, 1 \le x \le P \quad \textbf{(5)}$$

The proposed LB approach needs one more factor, **Load** for its scheduling rule. **Load** is defined as the wafer lots limited to machine m multiplied by the remaining layers of photolithography stage these wafer lots have. **Load** is a relative parameter, representing the load of the machine and wafer lots limited to one machine compared to other machines. The larger load factor means that the more required service from wafer lots has been limited to this machine. The definition and formula of the factor **Load** is as follows:

$$Load\ (m_x, s_j) = \sum_{t_i \in Wafer\ lots} \{t_i * R(t_i) \mid pm(t_i) = m_x\} \quad \textbf{(6)}$$

$$R(t_i) = \sum_{j=current}^{end\ step} Matrix\,[t_i, s_j] = r_2^x, 1 \le x \le p,$$

p is the numbers of photolithography machine

$pm(t_i)$ is the of dedicated photolithography machine x of t_i

The LB approach uses these factors to schedule the wafer lot to a suitable machine at the first photolithography stage which is the only photolithography stage without the dedicated constraint. Suppose we are currently at s_j, and the LB system will start from photolithography machine. We check if there is any wafer lot which is waiting for the photolithography machines at the first photolithography stage. The LB approach will assign the m_x with smallest ***Load(*m_x*, *s_j*)*** (formula (6)) for them one by one. After that, these wafer lots have been dedicated a photolithography machine. For each m_x, the LB approach will select one of the wafer lots dedicated to m_x which has the largest ***WaitStep(*t_i*)*** (formula (3)) for it. ***Load(*m_x*, *s_j*)*** of m_x will be updated after these two processes. The other wafer lots dedicated to each m_x which can not be allocated to the m_x at current step s_j will insert a w_2 for them in their pattern. For example, at the step s_{11}, t_i has been assigned to m_1, therefore, t_{i+1} will have a w_2 being inserted into at s_{11}, and then all the following required resource of t_{i+1} will shift one step The following matrix shows the situation.

	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	..	..	s_j	..	s_m
					..	..					
t_i	..	$\boxed{\mathbf{r_2}^{1}}$	$\mathbf{r_4}$	r_5	r_6	r_7	..	..			
t_{i+1}'	..	$\boxed{\mathbf{w_2}}$	$\mathbf{r_2}^{1}$	$\mathbf{r_4}$	r_6	r_5	..	..	..	..	
..		↑	→	→	→	→					

All the other types of machine will have same process without need of being concerned with the dedicated machine constraint. Therefore, we assigned one of the wafer lots which has the largest ***WaitStep(*t_i*)***, then the second largest one, and so on for each machine r_k. The LB approach will insert a w_k for the wafer lots do not be assigned to machines r_k at current step. Therefore, ***WaitStep(*t_i*)*** is to represent the delay status of t_i.

Simulation Result

We have done two types of simulations for a Least Slack (LS) time scheduling policy and our proposed LB approach. The LS time scheduling has been developed in the research, Fluctuation Smoothing Policy for Mean Cycle Time (FSMCT) (Kumar and Kumar 2001) in which the FSMCT is for re-entrant production lines. The entire class of LS policies has been proven stable in a deterministic setting (Lu and Kumar 1991) (Kumar 1994). The LS scheduling policy sets the highest priority to a wafer lot whose slack time is the smallest in the queue buffer of one machine. When the machine is going to idle, it will select the highest priority wafer lot in the queue buffer to service next. However, the simulation result shows that the proposed LB approach is better than the LS method. For simplifying the simulation to easily represent the scheduling methods, we have made the following assumptions:

(1) Each wafer lot has the same process steps and quantity.

(2) There is no breakdown event in the simulations.

(3) There is unlimited capacity for non-photolithography machines.

We implemented a simulation program in Java and have run the simulations on the NetBeans IDE 4.1. Our simulation was set with four to eleven photolithography machines and 500 wafer lots. There are 139 steps and 15 photolithography stages for each wafer lot in the simulations. The task pattern is as follows:

$rrrrr_2rrrrrrrrrrrrrrr_2r_2r_2r_2r_2r_2rrrrrrrrrrrrrrr_2r_2r_2r_2rrrrrrrr$
$rrrrrrrrrr_2r_2r_2rrrrrrrr_2r_2rrrrrrr_2r_2rrrrr_2r_2rrrrr_2r_2rrrrr_2r_2rrr$
$rr_2r_2rrrrr_2r_2rrrrr_2r_2rrrrr_2r_2rrrrr_2r_2rrrrr_2r_2rrrr$

r represents the non-photolithography stage; and single or consecutive r_2 the photolithography stage. The wafer lot t_2 starts to process at s_3 when t_1 has passed two steps. t_3 starts at s_4 when t_1 has passed three steps . We simulate the wafer arrival rate between two wafer lots as a Poisson distribution.

Tasks Matrix

$s_1s_2s_3s_4$…………………………………………………… s_m

t_1: $rrrrr_2rrrrrrrrrrrrrrr_2r_2r_2r_2r_2r_2rrrrrr$……

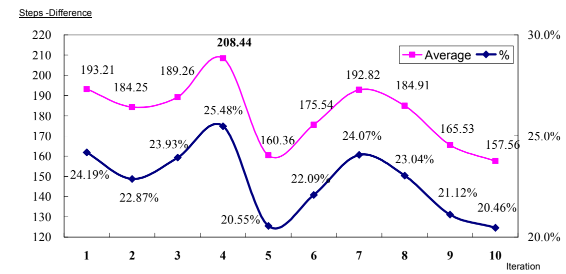
t_2: $rrrrr_2rrrrrrrrrrrrrrr_2r_2r_2r_2r_2r_2rrrrrr$……

t_3: $rrrrr_2rrrrrrrrrrrrrrr_2r_2r_2r_2r_2r_2rrrrrr$……

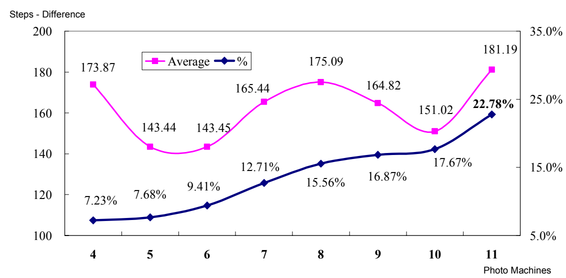
:

t_{500}: :

We applied the LS and LB methods for these photolithography machines to select the next wafer lot to process in the simulations. When the wafer lot needs to wait for its dedicated machine, we insert a “*w*” in the task pattern of the wafer lot to represent the situation. After completing the simulations, we count the pattern of wafer lots to obtain how much time they have used. The simulation result is shown in Figure 3.



(a) Simulations Result - 11 Photo Machines



(b) Simulation Result - Diff. Photo. Machines

Figure 3. Simulation Result

By comparing the mean of cycle time of the case using 11 machines and ten iterations, the LS method has an average 181.19 steps more than the LB method. In other words, the LB method is better than the LS method 22.78% on average in the simulation. The simulations result of different photolithography machines indicates that the more the photolithography machines, the better the LB method performs than the LS method does.

Although the simulations are simplified, they reflect the real situation we have met in the factory. After applying the LS method to the above simulations and counting the required resource (formula (5) $k=2$, $x = 4$ *to* 11) for the photolithography machines at each step we can realize that the load of these machines become unbalanced during the simulations. It is not difficult to extend the simulation with more machines, wafer lots, and stages.

Conclusion

To provide the solution to the issue of dedicated machine constraint, the proposed Load Balancing (LB) scheduling approach has been presented. Along with providing the LB scheduling approach to the dedicated machine constraint, we also presented a novel model—the representation and manipulation method for the task patterns. The simulations also showed that our proposed LB scheduling approach was better than the LS method. The advantage of LB is that we could easily schedule the wafer lots by simple calculation on a two-dimensional matrix. Moreover, the matrix architecture is easy for practicing other semiconductor manufacturing problems in the area with a similar constraint. We also want to apply other scheduling rules to the Resource Allocation module of the RSEM. To develop a knowledge-based scheduling system for the Resource Allocation module or to model the module as distributed constraint satisfaction scheduling project are our intended future work.

References

Akcalt, E.; Nemoto, K.; and Uzsoy, R. 2001. Cycle-time improvements for photolithography process in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing* 14(1): 48-56.

Arisha, A., and Young, P. 2004. Intelligent Simulation-based Lot Scheduling of Photolithography Toolsets in a Wafer Fabrication Facility. In *Winter Simulation Conference*, 1935-1942.

Chern, C., and Liu, Y. 2003. Family-Based Scheduling Rules of a Sequence-Dependent Wafer Fabrication System. *IEEE Transactions on Semiconductor Manufacturing* 16(1):15-25.

Kumar, P. R. 1993. Re-entrant Lines. In *Queuing Systems: Theory and Applications, Special Issue on Queuing Networks* 13(1-3): 87-110.

Kumar, P. R. 1994. Scheduling Manufacturing Systems of Re-Entrant Lines. *Stochastic Modeling and Analysis of Manufacturing Systems*, 325-360. David D. Yao (ed.), New York: Springer-Verlag.

Kumar, S., and Kumar, P. R. 2001. Queuing Network Models in the Design and Analysis of Semiconductor Wafer Fabs. *In IEEE Transactions on Robotics and Automation* 17(5):548-561.

Lu, S. H., and Kumar, P. R. 1991. Distributed Scheduling Based on Due Dates and Buffer Priorities. *IEEE Transactions on Automatic Control* 36(12):1406-1416.

Mönch, L.; Prause, M.; and Schmalfuss V. 2001. Simulation-Based Solution of Load-Balancing Problems in the Photolithography Area of a Semiconductor Wafer Fabrication Facility. In *Winter Simulation Conference*, 1170-1177.

Shr, A. M. D.; Liu, A.; and Chen, P. P. 2006a. A Load Balancing Scheduling Approach for Dedicated Machine Constraint. In *ICEIS2006, Proceedings of the 8th International Conference on Enterprise Information Systems,* Paphos, Cyprus, May 2006. Forthcoming.

Shr, A. M. D.; Liu, A.; and Chen, P. P. 2006b. A Heuristic Load Balancing Scheduling Method for Dedicated Machine Constraint. In *Proceedings of the International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE'06)*, Annecy, France, June 2006. Forthcoming.