Scheduling for Dedicated Machine Constraint

Arthur Shr[1], Peter P. Chen[1] and Alan Liu[2]
[1]Department of Computer Science, Louisiana State University,
[2]Department of Electrical Engineering, National Chung Cheng University
[1]U.S.A., [2]Taiwan, R.O.C.

1. Introduction

We have proposed the heuristic Load Balancing (LB) scheduling (Shr et al., 2006a) (Shr et al., 2006b) (Shr et al., 2006c) and Multiagent Scheduling System (MSS) (Shr, et al. 2006d) approaches to provide solutions to the issue of dedicated photolithography machine constraint. The dedicated photolithography machine constraint, which is caused by the natural bias of the photolithography machine, is a new challenge in the semiconductor manufacturing systems. Natural bias will impact the alignment of patterns between different layers. This is especially true for smaller dimension IC for high technology products. A study considered different production control policies for semiconductor manufacturing, including a "machine dedication policy" in their simulation, has reported that the scheduling policy with machine dedication had the worst performance of photolithography process (Akcalt et al., 2001). The machine dedication policy reflects the constraint we are discussing here.

In our previous work, along with providing the LB scheduling or MSS approaches to the dedicated machine constraint, we have also presented a novel model––the Resource Schedule and Execution Matrix (RSEM) framework. This knowledge representation and manipulation method can be used to tackle the dedicated machine constraint. A simulation system has also been implemented in these researches and we have applied our proposed scheduling approaches to compare with the Least Slack (LS) time approach in the simulation system (Kumar & Kumar, 2001). The reason for choosing the LS scheduling approach was that this approach was the most suitable method for solving the types of problems caused by natural bias at the time of our survey.

The LS scheduling approach has been developed in the research of Fluctuation Smoothing Policy for Mean Cycle Time (FSMCT) (Kumar & Kumar, 2001), in which the FSMCT scheduling policy is for the re-entrant production lines. The entire class of the LS scheduling policies has been proven stable in a deterministic setting (Kumar, 1994) (Lu & Kumar, 1991). The LS approach sets the highest priority to a wafer lot whose slack time is the smallest in the queue buffer of one machine. When the machine becomes idle, it selects the highest priority wafer lot in the queue buffer to service next. However, the simulation result has shown that the performances of both our proposed LB and MSS approaches were better than the LS method. Although the simulations were simplified, they have reflected the real situation we have met in the factory.

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

Extending the previous simulations, we introduce two different types of simulation for the dedicated machine constraint in this paper. One is to show that our proposed LB scheduling approach is still better than the LS approach under the different capacity and service demand of the wafer lots. The case of setting with different photolithography machines represents the different capacity of the semiconductor factory, while the case of setting with different photolithography layers represents the different products' demand for the semiconductor factory. The other simulation is to show the situation of the thrashing phenomenon, i.e., the load unbalancing among the photolithography machines during the process when we apply the LS approach. We have also learned that the load unbalancing is consistent with different photolithography machines.

The rest of the paper is organized as follows: Section 2 describes the motivation of this research including the description of dedicated machine constraint, the load balancing issue, and related research. In Section 3, we present the construction procedure and algorithms of the RSEM framework to illustrate the proposed approach for dedicated machine constraint. The proposed LB scheduling approach is presented along with an example of the semiconductor factory in Section 4. Section 5 shows the simulation results and we conclude the work in Section 6.

2. Motivation

2.1 Dedicated Machine Constraint

Dedicated machine constraint forces wafer lots passing through each photolithography stage 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. Fig. 1. describes the dedicated machine constraint. When material enters the photolithography stage with dedicated machine constraint, the wafer lots dedicated to machine X need to wait for it, even if machine Y is idle. By contrast, when wafer lots enter into non-photolithography stages without any machine constraints, they can be scheduled to any machine, A, B, or C.

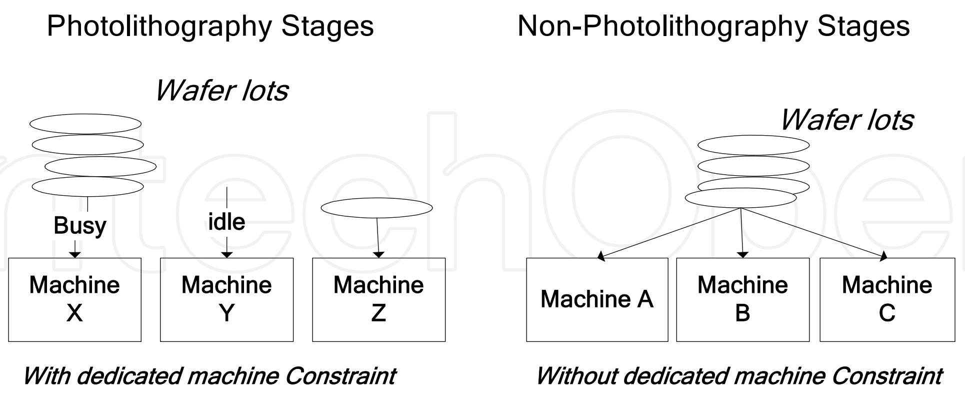


Figure 1. Dedicated machine constraint

Presently, the dedicated machine constraint is the most significant barrier to improving productivity and fulfilling the requests of customers. It is also the main contributor to the complexity and uncertainty of semiconductor manufacturing. Moreover, photolithography is the most important process in semiconductor manufacturing. A good yield of IC products is heavily dependent on a good photolithography process. At the same time, the process can also cause defects. Therefore, the performance of a factory particularly relies on the performance of photolithography machines.

2.2 Load Balancing Issue

The load balancing issue is mainly derived 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 all subsequent photolithography stages. Therefore, 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. Any unexpected abnormal events or a breakdown of machines will cause a pile-up of many wafer lots waiting for the machine and cause a big problem for the factory. Therefore, the unbalanced load among photolithography machines means that some of the photolithography machines become idle and remain so for a while, due to the fact that no wafer lots can be processed, and the other is always busy while many wafer lots bound to this machine are awaiting processing. As a result, some wafer lots are never delivered to the customer on time, and the performance of the factory decreases. Moreover, it cannot meet the fast-changing market of the semiconductor industry.

2.3 Related Research

The scheduling problems of the semiconductor manufacturing systems or photolithography machines have been studied by some researchers. By using a queuing network model, a "Re-Entrant Lines" model has been proposed to provide the analysis and design of the semiconductor manufacturing system. Kumar's research described several scheduling policies with some results concerning their stability and performance (Kumar, 1993) (Kumar, 1994). These scheduling policies have been proposed to deal with the buffer competing problem in the re-entrant production line, wherein they pick up the next wafer lot in the queue buffers when machines become idle. A study proposed a stochastic dynamic programming model for scheduling a new wafer lot release and bottleneck processing by stage in the semiconductor factory. This scheduling policy is based on the paradigm of stochastic linear quadratic control and incorporates considerable analysis of uncertainties in products' yield and demand (Shen & Leachman, 2003). A special family-based scheduling rule, Stepper Dispatch Algorithm (SDA-F), is proposed for the wafer fabrication system (Chern & Liu, 2003). SDA-F uses a rule-based algorithm with threshold control and least slack principles to dispatch wafer lots in photolithography stages. Many queuing network scheduling policies or methods have been published to formulate the complexity of semiconductor manufacturing problems; however, they need to be processed off-line and cannot respond rapidly to dynamic changes and uncertainty in the environment.
Vargas-Villamil, et al. proposed a three-layer hierarchical approach for semiconductor reentrant manufacturing (Vargas-Villamil et al., 2003), which decomposes the big and intractable problems of semiconductor manufacturing into smaller control problems. It

reduces the effort and frequency of the control decisions. 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. Two approaches were reported to use simulations to model the photolithography process. One of them proposed a Neural Network approach to develop an intelligent scheduling method according to a qualifying matrix and lot scheduling criteria to improve the performance of the photolithography machines (Mönch et al., 2001). The other approach decides the wafer lots assignment of the photolithography machines at the time when the wafer lots are released to the manufacturing system in order to improve the load-balancing problem (Arisha & Young, 2004). These researches have emphasized that photolithography process scheduling issues are the most important and critical challenge of the semiconductor manufacturing system. However, it might be difficult to have the proper training data to build a Neural Network scheduling system. It is also inefficient to manually adjust lot scheduling criteria or lot assignment to fit the fast-changing market of semiconductor manufacturing. Moreover, their proposed scheduling methods did not concern the dedicated machine constraint.

3. Resource Schedule and Execution Matrix (RSEM) Framework

In this section, the procedure and algorithm associated with the Resource Schedule and Execution Matrix (RSEM) framework are presented. The RSEM framework construction process consists of three modules including the *Task Generation*, *Resource Calculation*, and *Resource Allocation* modules.

The first module, *Task Generation,* models the tasks for the scheduling system and it is represented in a two-dimensional task matrix. One dimension is reserved for the tasks, t_1, t_2, …, t_n; the other represents the periodical time events (or steps) s_1, s_2, …,s_m. Each task has a sequential Process Pattern to represent the resources it needs to go from the raw material to a product during the process sequence and we put the process pattern in an array. We define each type of resource as r_k, $k = 1$ to o. For example, the process pattern, r_1, r_2, …, r_o, means that a particular task needs the resources in the sequence of r_1 first and r_2 following that until r_o is gained. Therefore, the matrix looks as follows:

$$\begin{array}{c|ccccccccccc}
 & s_1 & s_2 & . & . & s_q & . & . & . & s_j & . & s_m \\
\hline
t_1 & r_1 & r_2 & r_3 & .. & .. & .. & ... & .. & .. & .. & .. \\
t_2 & & r_3 & r_4 & .. & .. & .. & .. & .. & .. & .. & .. \\
. & & & & r_1 & r_3 & .. & .. & .. & .. & & \\
t_i & & & & & r_3 & r_4 & .. & .. & r_k & .. & \\
. & & & & & & & . & . & & & \\
t_n & & & & & .. & .. & .. & .. & .. & .. &
\end{array}$$

The symbol r_k in the task matrix entry [t_i, s_j] represents that task t_i needs the resource r_k at the time s_j. If t_i starts to be processed at s_q, and the total number of steps needed for t_i is p, we will fill its process pattern into the matrix from [t_i, s_q]… to [t_i, s_{q+p-1}] with r_k, $k = 1$ to o. All the tasks, $t_1 \ldots t_n$, follow the illustration above to form a task matrix in the *Task Generation* module. To represent dedicated machine constraint in the matrix for this research, the symbol r_k^x, a replacement of r_k, represents that t_i has been dedicated to a particular instance x of a resource type r_k at s_j. One more symbol w_k represents the wait situation when the r_k cannot allocate to t_i at s_j. The situation can be that r_k is assigned to other higher priority tasks or it is breakdown. This symbol will be used in the *Resource Allocation* module.
The *Resource Calculation* module summarizes the value of each dimension as the factors for the scheduling rule of the *Resource Allocation* module. For example, by counting the task pattern of the row t_i in the task matrix, we can determine how many steps t_i processed after it finished the whole process. We can also realize how many wait steps t_i has had by counting w_k from the starting step to the current step in that row of the task matrix. Furthermore, if we count the symbol r_k^x at the column s_j, we can know how many tasks will need the machine m_x of resource r_k at s_j.
We need to generate the task matrix, obtain all the factors for the scheduling rules, and build up the scheduling rules before starting the execution of the *Resource Allocation* module. The module schedules 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 the resource of r_k is not available for t_i at s_j, then we will not only insert the symbol w_k in the pattern of t_i , but will also need to shift one step for the process pattern following t_i in the matrix. Therefore, we can obtain the updated factor for the number of tasks waiting for r_k at s_j by simply counting w_k at the column s_j. We can also obtain the factor for the number of wait steps t_i has by counting w_k, $1 \leq k \leq o$ by the row t_i in the matrix.
Our proposed approach can provide two kinds of functions. One is that, to define the factors and resource allocation rules according to expert knowledge, we can quickly determine the allocation of resources at each step by the factors summarized from the task matrix. The other is that we can 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. Moreover, by using different predefined rules and factors, the RSEM framework could apply to different scheduling issues or constraints of semiconductor manufacturing.

3.1 Procedure for Constructing the RSEM framework

To better understand our proposed scheduling process, the flowchart of the RSEM framework construction process is shown in Fig. 2. The process of using the RSEM framework 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 the scheduling process is in the *Resource Allocation* module. When we have scheduled 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 arrives in the system.

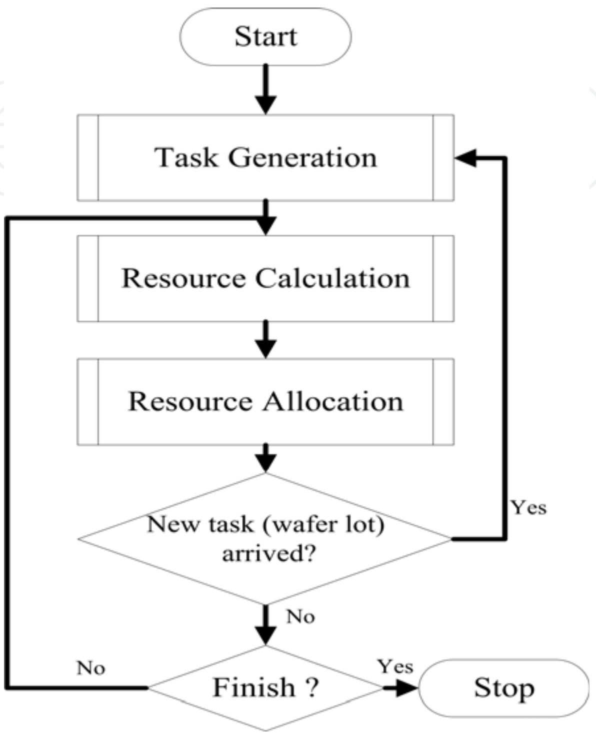


Figure 2. Flowchart of the RSEM framework construction process

3.2 Algorithms Associated to the RSEM framework

To make the construction process of the proposed RSEM framework 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 task matrix by copying the task patterns of the tasks in the matrix. It will start from the start step s_s and go to the end step s_e of each task. The s_s will not start before the current step s_c and the s_e should not end beyond the maximum step m of the matrix in the system. The task matrix will be passed to and manipulated at the other two algorithms.

```
Algorithm-1 **Task_Generation**
{
// $s_c \le s_s \le s_e \le m$, where $m$ is the max step in system, and
// $s_c$ is current step.

   **for** $i = 1$ **to** $n$ **do**
        Copy task pattern of $t_i$ into matrix from its starting step $s_s$, to its ending step $s_e$
    **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 $n$ do
        for $j$ = 1 to $m$ do;
              if (matrix[$t_i$,$s_j$] is not empty) then
                    Total_Step($t_i$)= Total_Step($t_i$) + 1; /*Count total steps*/
              end if
        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 $n$ do
        for $j$ = 1 to $s_c$ do;
              if (matrix[$t_i$,$s_j$] = $w_k$) then
                    Wait_Step($t_i$)= Wait_Step($t_i$) + 1; /*Count wait steps*/
              end if
        next
  next
//Factor for resource, function: Resource_Demand($r_k$)
//To count total tasks which are need of the resource $r_k$
//$o$: total resource.
  for $k$ = 1 to $o$ do
        for $i$ = 1 to $n$ do
              if (matrix[$t_i$,$s_c$] = $r_k$) then
                    Resource_Demand($r_k$)= Resource_Demand($r_k$) + 1;
              end if
        next
  next
//Factor for Resource, function: Queue_Buffer($r_k$)
//To count total tasks which are waiting for of the resource $r_k$
  for $k$ = 1 to $o$ do
        for $i$ = 1 to $n$ do
              if (matrix[$t_i$,$s_c$] = $w_k$) then
                    Queue_Buffer($r_k$)=+1;
              end if
        next
  next

//Factor for ...
…. // factor Load, Utilization, and so on.
}
```

We will have four factors ready for scheduling after the *Resource Calculation* process described in Algorithm-2, namely, **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 can also include other factors in this module depending on different applications, e.g., the factors of the load of a particular photolithography machine and the remaining photolithography stages of the tasks in the example of Section 3.
The procedure of Algorithm-3 executes the scheduling process for the tasks and resources. The first part of the scheduling process allocates 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 our proposed Load Balancing approach is one of the examples. After the process for resource 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 to a machine. A wait symbol w_k represents the state of waiting for machine type $\boldsymbol{k}$, and a w_k^x is waiting for dedicated machine number $\boldsymbol{x}$, m_x, of machine type $\boldsymbol{k}$.

```
Algorithm-3 Resource_Allocation
{
//Scheduling; $o$: total resource, $s_c$:current step.
  for $k = 1$ to $o$ do
        Assign tasks to $r_k$, according to predefined rules
        e.g., the Load Balancing scheduling (LB),
                  Multiagent Scheduling System (MSS) or
                  Least Slack time scheduling (LS) rules
  next

//Execution; shift process pattern of the tasks,
//which do not be scheduled at current step;
//$x$: the machine number.
  for $i = 1$ to $n$ do
      if ($t_i$ will not take the resource at this step) then
          insert $w_k$ to wait for $r_k$; /* without dedicated constraint */
          or
          insert $w_x^k$ to wait for $m_x$ of $r_k$; /*dedicated constraint */
      end if
  next
}
```

4. Load Balancing Scheduling Method

In this section, we apply the proposed Load Balancing (LB) scheduling method to the dedicated machine constraint of the photolithography machine in semiconductor manufacturing. The LB method uses the RSEM framework as a tool to represent the temporal relationship between the wafer lots and machines during each scheduling step.

4.1 Task Generation

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 the matrix representation. There exist thousands of wafer lots and hundreds of process steps in a typical factory. We start by transforming the process pattern of wafer lots into a task matrix. We let r_2 represent the photolithography machine and r_k $(k \neq 2)$ represent non-photolithography machines. The symbol r_2^x in the matrix entry $[i,j]$ represents the wafer lot t_i needing the photolithography machine m_x at the time s_j with dedicated machine constraint, while r_k $(k \neq 2)$ in $[i,j]$ represents the wafer lot t_i needing the machine type k at s_j without dedicated machine constraint. There is no assigned machine number for the photolithography machine before the wafer lot has passed the first photolithography stage. Suppose that the required resource pattern of t_1 is as follows: $r_1r_3r_2r_4r_5r_6r_7r_2r_4r_5r_6r_7r_8r_9r_1r_3r_2r_4r_5r_6r_7r_3r_2r_8r_9$…and starts the process in the factory at s_1. We will fill its pattern into the matrix from $[t_1,s_1]$ to $[t_1,s_n]$, which indicates that t_1 needs the resource r_1 at the first step, resource r_3 at the second step, and so on. The photolithography process, r_2, in this process pattern has not been dedicated to any machine and the total number of steps for t_1 is n. The task t_2 in the task matrix has the same process pattern as t_1 but starts at s_3; meanwhile, t_i in the matrix starts at s_8. It requires the same type of resource r_2, the photolithography machine, but the machine is different from the machine t_2 needed at s_{10}; i.e., t_2 needs the machine m_1, while t_i has not been dedicated to any machine yet. Two tasks, t_2 and t_i, might compete for the same resource r_4 at s_{11} if r_4 is not enough for them at s_{11}. The following matrix depicts the patterns of these tasks.

	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_{20}	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}	..	..	s_j	..	s_m
t_1	r_1	r_3	r_2	r_4	r_5	r_6	r_7	r_2	r_4	r_5	r_6	r_7	r_8	..	..	r_6	r_7	r_3	r_2	r_8	r_9	..	..			.
t_2			r_1	r_3	r_2	r_4	r_5	r_6	r_7	$\boldsymbol{r_2^1}$	$\boldsymbol{r_4}$	r_5	r_6	..	..	r_4	r_5	r_6	r_7	r_3	r_2	r_8	r_9	..	..	
..																										
t_i								r_1	r_3	$\boldsymbol{r_2}$	$\boldsymbol{r_4}$	r_6	r_5	..	..	..	..	..	..	..	..	..	..	$\boldsymbol{r_k}$	..	

4.2 Resource Calculation

The definitions and formulae of these factors for the LB scheduling method in the *Resource Calculation* module are as follows:

W: wafer lots in process,

W_p : wafer lots dedicated to the photolithography machine, p,

P: numbers of photolithography machines,

$R(t_i)$: remaining photolithography layers for the wafer lot t_i,

K: types of machine (resource),

s_s: start step, s_c: current step, s_e: end step.

Required resources:

- How many wafer lots will need the k type machine x at s_j?

$$RR(r_k^x, s_j) = \sum_{t_i \in W} \{t_i \mid [t_i, s_j] = r_k^x\},\ 1 \le x \le P \tag{1}$$

Count steps:

- *How many wait steps did t_i have before s_j?*

$$WaitStep(t_i) = \sum_{j=s_s}^{s_c} \{t_i \mid [t_i, s_j] = w_k\},\ 1 \le k \le K \tag{2}$$

- *How many steps will t_i have?*

$$Steps\,(t_i) = \sum_{j=s_s}^{s_e} \{t_i \mid [t_i, s_j] \neq null\} \tag{3}$$

- *The load factor, L_p, of the photolithography machine p.*

$$L_p = \sum_{t_i \in W_p} \{t_i \times R(t_i)\} \tag{4}$$

L_p is defined as the wafer lots that are limited to machine p multiplied by the remaining layers of photolithography stages these wafer lots have. L_p is a relative parameter, representing the load of the machine and wafer lots limited to one machine compared to other machines. A larger L_p means that more required service from wafer lots is limited to this machine. The LB scheduling method 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.

4.3 Resource Allocation

The process flow of the *Resource Allocation* module for the example is described in this section. Suppose we are currently at s_j, and the LB scheduling method will start from the photolithography machine. We check to determine if there is any wafer lot that is waiting for the photolithography machines at the first photolithography stage. The LB method will assign the p with the smallest L_p for them, one by one. After that, these wafer lots will be dedicated to a photolithography machine. For each p, the LB method will select one wafer lot of W_p that has the largest WaitStep(t_i) for it. The load factor, L_p, will be updated after these two processes. The other wafer lots dedicated to each p, which cannot be allocated to the p at current step s_j, will insert a w_2 in their pattern. For example, at s_{10}, t_i has been assigned to p; therefore, t_{i+1} will have a w_2 inserted into s_{10}, and then all the following required resources of t_{i+1} will shift one step. All other types of machines will have the same process without need to be concerned with the dedicated machine constraint. Therefore, we assigned the wafer lot that has the largest WaitStep*(t_i)*, then the second largest, and so on for each machine r_k. Similarly, the LB method will insert a w_k for the wafer lots which will not be assigned to machines r_k at this current step. Therefore, WaitStep(t_i) represents the delay status of t_i.

	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	..	..	s_j	..	s_m
..					..	..					
t_i	..	r_2^p	r_4	r_6	r_5	r_7	..	..			
t_{i+1}	..	w_2	r_2^p	r_4	r_6	r_5	..	..	..	..	
..		$\uparrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$					

4.4 Discussion

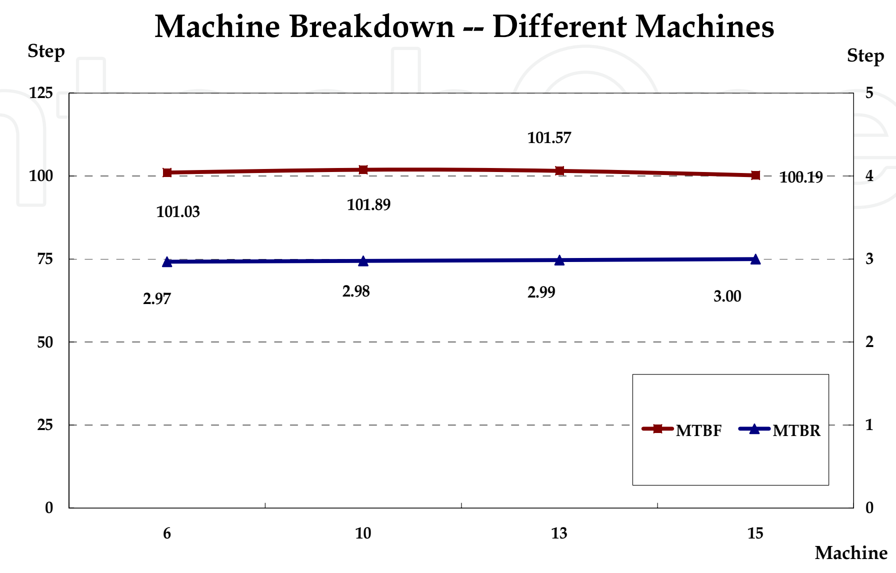
Realistically, it is not difficult to approximate the real machine process time for different steps using one or several steps together with a smaller time scale step. We can also use the RSEM framework to represent complex tasks and allocate resources by simple matrix calculation. This reduces much of the computation time for the complex problem.
Another issue is that the machines in the factory have a capacity limitation due to the capital investment, which is the resource constraint. The way to make the most profit for the investment mostly depends on optimal resource allocation techniques. However, most scheduling policies or methods can provide neither the exact allocation in an acceptable time, nor a robust and systematic resource allocation strategy. We use the RSEM framework to represent complex tasks and allocate resources by simple matrix calculation. This reduces much of the computation time for the complex problem.

5. Simulation

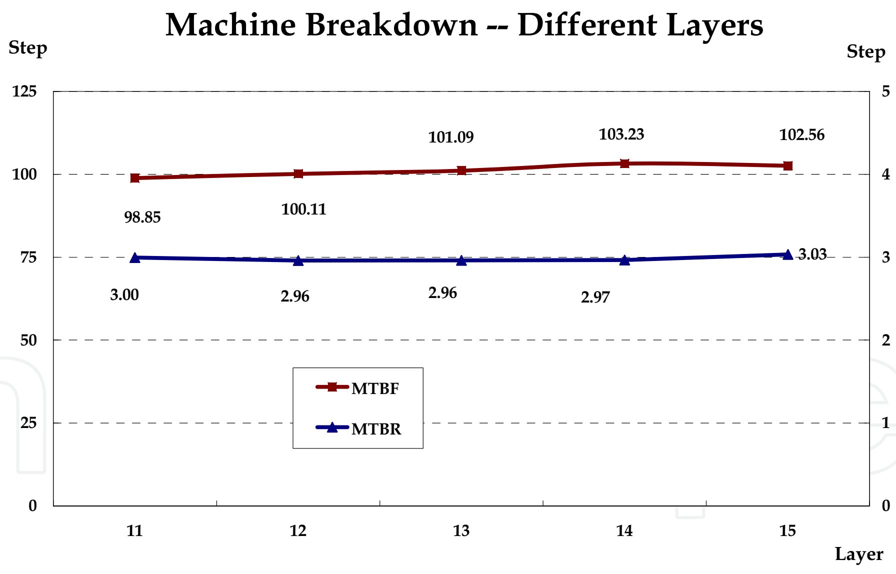
We have done two types of simulations using both the Least Slack (LS) time scheduling and our LB approach. The LS policy has been developed in the research, Fluctuation Smoothing Policy for Mean Cycle Time (FSMCT) (Kumar & Kumar, 2001). The FSMCT scheduling policy is for re-entrant production lines. 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 becomes idle, it will select the highest priority wafer lot in the queue buffer to service next. The entire class of LS policies has been proven stable in a deterministic setting (Kumar, 1994) (Lu & Kumar, 1991), but without the dedicated machine constraint. To simplify the simulation to easily represent the scheduling approaches, we have made the following assumptions: (1) each wafer lot has the same process steps and quantity, and (2) there is an unlimited capacity for non-photolithography machines

5.1 Simulation Results

We implemented a simulation program in Java and ran the simulations on NetBeans IDE 5 (http://www.netbeans.org/). To represent the different capacity and required resource demand situation for a semiconductor factory, we take account of different photolithography machines and wafer lots with different photolithography layers in the simulation program. Our simulation was set with 6, 10, 13, and 15 photolithography machines, and 11 to 15 photolithography layers. There are 1000 wafer lots in the simulation. The wafer arrival rate between two wafer lots is a Poisson distribution. We also set up the probability of breakdown with 1% for each photolithography machine at each step in the simulation. The duration of each breakdown event may be 1 to 4 steps and their individual probability is based on a Uniform distribution.
Fig. 3(a) illustrated the average Mean Time Between Failures (MTBF) and Mean Time Between Repairs (MTBR) of different photolithography machines, i.e. in the case of 6 machines the average of MTBF and MTBR is 101.03 and 2.97 steps, respectively. While the average MTBF and MTBR of different photolithography layers are shown in Fig. 3(b), in the case of 15 layers the average of MTBF and MTBR is 102.56 and 3.30 steps, respectively.



(a)



(b)

Figure 3. MTBF and MTBR of machine breakdown

In the task patterns, the symbol *r* represents the non-photolithography stage; and r_2 the photolithography stage. The basic task pattern for 11 layers is: “*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrrrrrrrrrrrrr*$r_2r_2r_2$*rrrrrrr*r_2r_2*rrrrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrr*”. Then the task pattern for each added layer after 11 layers is: “r_2*rrrr*”. Therefore, the task pattern for 12 layers is: “*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrrrrrrrrrrrrr*$r_2r_2r_2$*rrrrrrr*r_2r_2*rrrrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrr*”,…, and the task pattern for 15 layers is: “*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrrrrrrrrrrrrr*$r_2r_2r_2$*rrrrrrr*r_2r_2*rrrrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrrr*r_2r_2*rrr*”. The task matrix for 15 layers looks as follows:

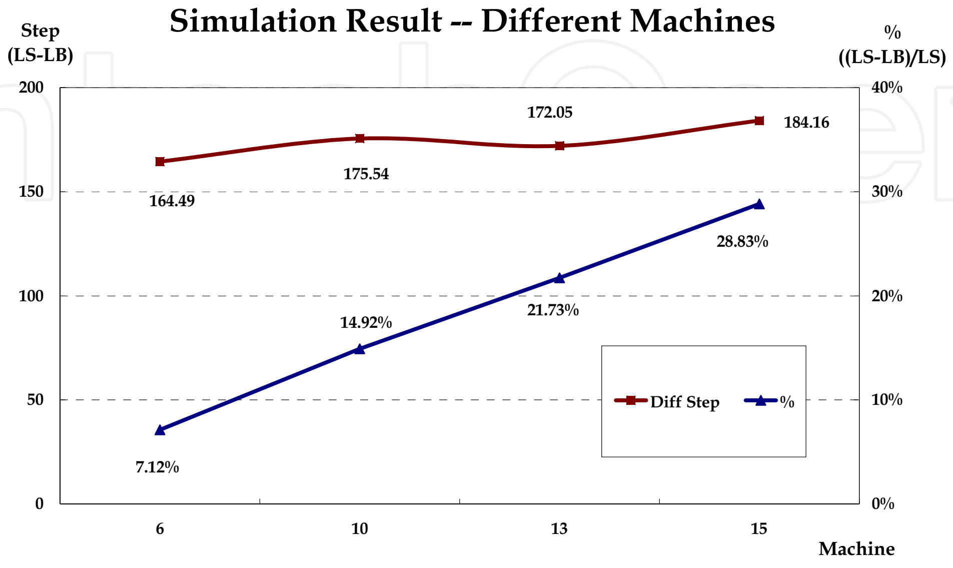
	s_1 ……………………………………………………………………………… s_m
t_1	*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrr*……..r_2r_2*rrrr*
t_2	*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrr*……..r_2r_2*rrrr*
..	:
t_i	*rrrr*r_2*rrrrrrrrrrrrrr*$r_2r_2r_2r_2r_2r_2$*rrrrrrrrrrrrrr*$r_2r_2r_2r_2$*rrrrr*……..
..	
t_{1000}	:

Task Matrix (15 photolithography layers)

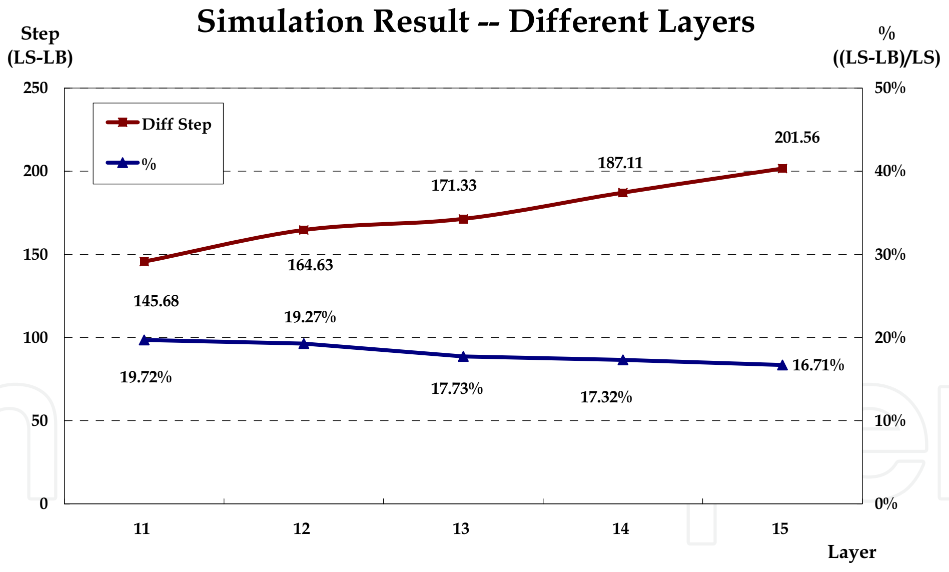
Both LB and LS approaches were applied to the same task matrix during each simulation generated by the *Task Generation* module described in Section 3. The result of simulation, as described in detail in the following subsections, shows the advantage of the LB approach over the LS approach under different numbers of machines by the average of the different photolithography layers, and under different numbers of layers by the average of the different photolithography machines.

Different Photolithography Machines: By comparing the mean of cycle time, in the case of 6 machines, the LS method has 164.49 (LS-LB) steps more than the LB approach. That is 7.12% ((LS-LB)/LS) more in the simulation. The different steps from machines 10 to 15 incrementally rise from 175.54 to 184.16 steps and the percentages of the difference rises from 14.92% to 28.83%. The simulation result of different photolithography machines indicates that the more photolithography machines, the better the LB approach performs than the LS method does**.** The simulation result is shown in Fig. 4(a).

Different Photolithography Layers: On the other hand, the simulation result of different layers (11 to 15 layers) indicates that there is no significant difference with different photolithography layers. The outperformance in percentage of the LB approach is between the minimum, 16.71%**,** to the maximum, 19.72%. Such a simulation result is shown in Fig. 4(b).



(a). Different Photolithography Machines



(b) Different Photolithography Layers

Figure 4. Simulation results

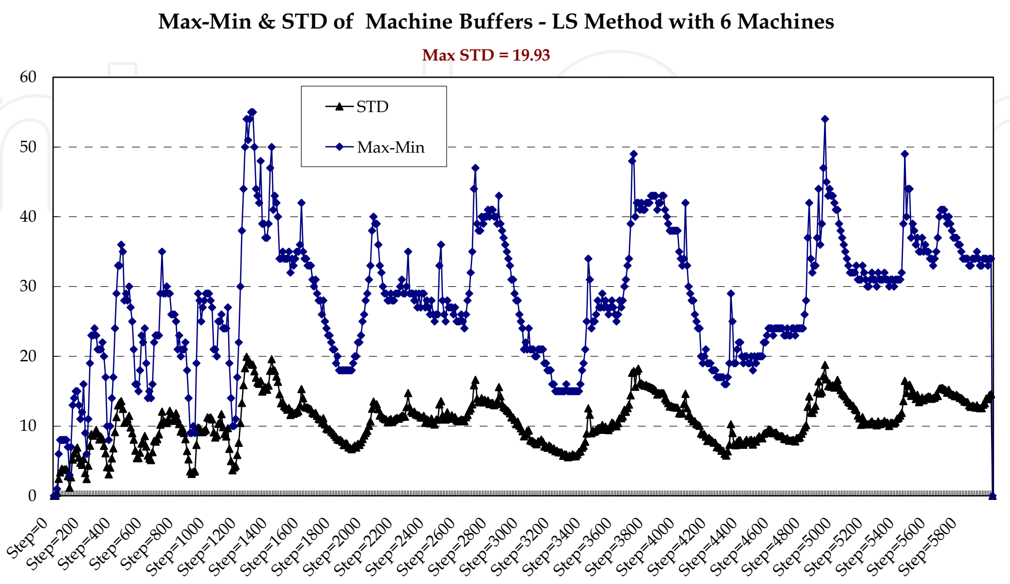
5.2 Thrashing and Load Unbalancing

After applying the LS approach to the above simulations and counting the required resource (Equation (1): RR(r_k^x, s_j) of Section 4.2, k=2, x=6, 10, 13, 15) for the photolithography machines at each step, we can observe that the load balancing status in terms of the difference between maximum and minimum counts of the required resource for the machines becomes larger and unstable, i.e., the thrashing phenomenon takes place in the simulations. The simulation is set with 6, 10, 13, and 15 photolithography machines and 15 photolithography layers. In the simulation, the Max-Min and Standard Deviation of wafer lots in machine buffers set with 6, 10, 13, and 15 machines are shown in the graphs in Fig. 5 ((a), (b), (c), and (d)), respectively. The simulation results also reveal that the fewer machines the system has, the worse the situation of an unbalanced load would be to the system. On the other hand, while applying the LB method, the load balancing status is stable and consistent with different machines, and it is always less then 2.5 at each execution step.

6. Conclusion

We presented the Resource Schedule and Execution Matrix (RSEM) framework–a novel representation and manipulation method for the tasks in this paper. The RSEM framework is used as a tool to analyze the issue of dedicated machine constraint and develop solutions. The simulation also showed that the proposed LB scheduling approach was better than the LS method in various situations. Although the simulations are simplified, they reflect the real situation we have met in the factory.

The advantage of the proposed RSEM framework is that we can easily apply various policies to the scheduling system by simple calculation on a two-dimensional matrix. 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 in the RSEM framework. Our intended future work is to develop a knowledge-based scheduling system for the *Resource Allocation* module or to model it as distributed constraint satisfaction scheduling project.



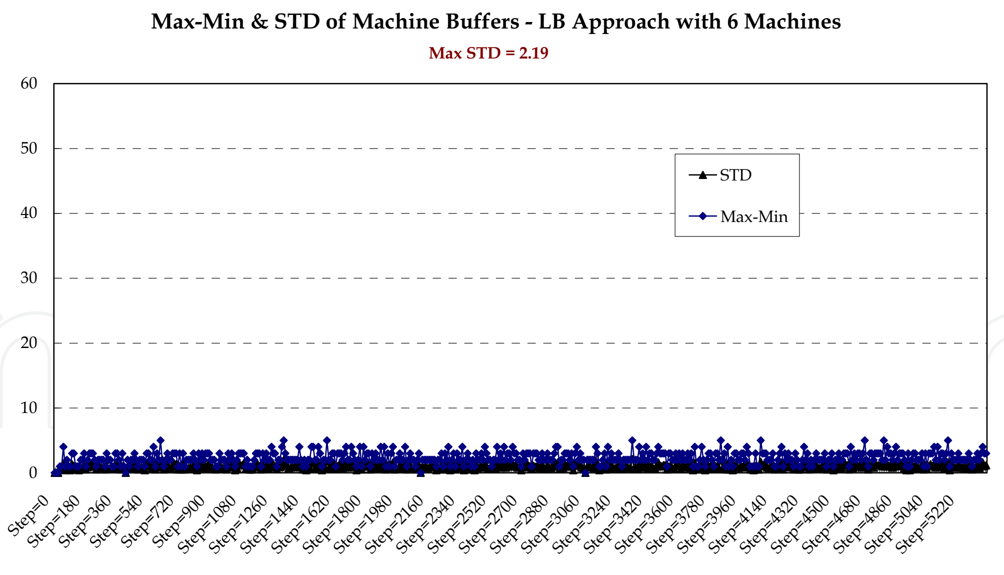
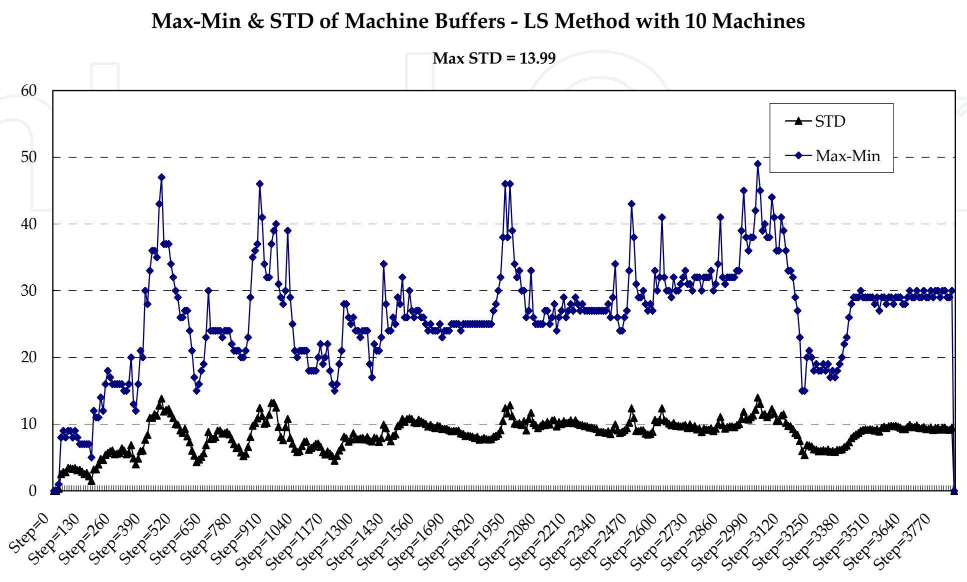


Figure 5(a). Thrashing phenomenon—6 machines



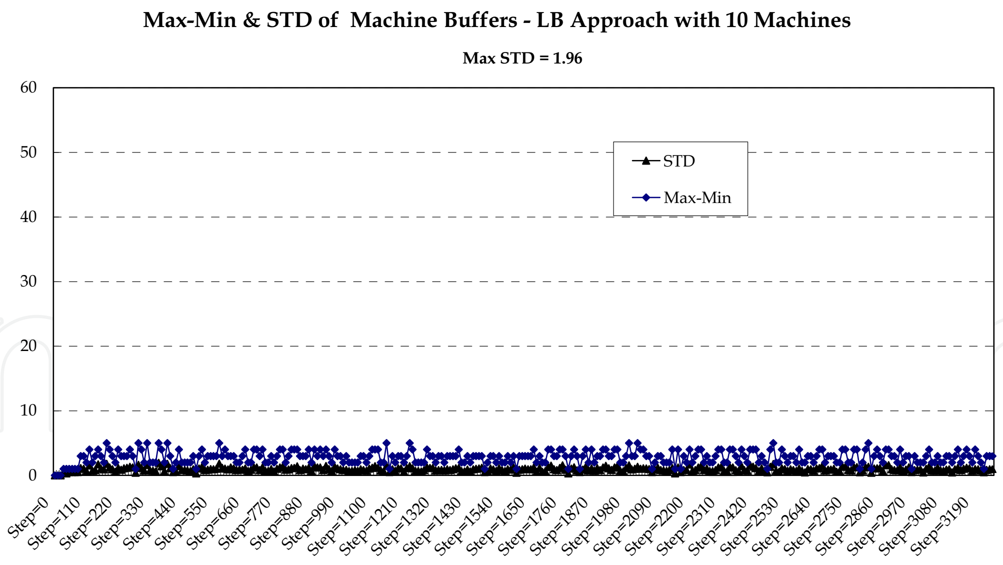
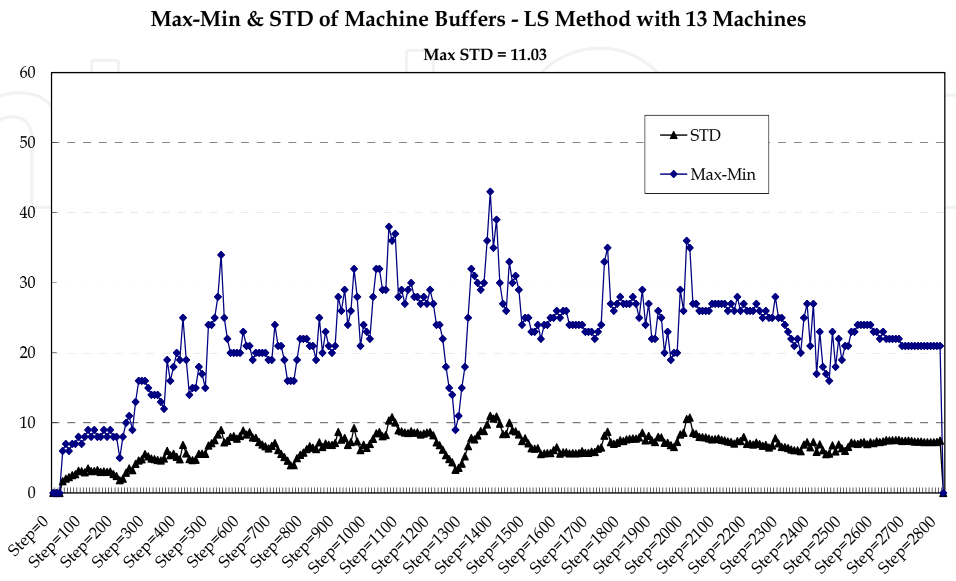


Figure 5(b). Thrashing phenomenon—10 machines



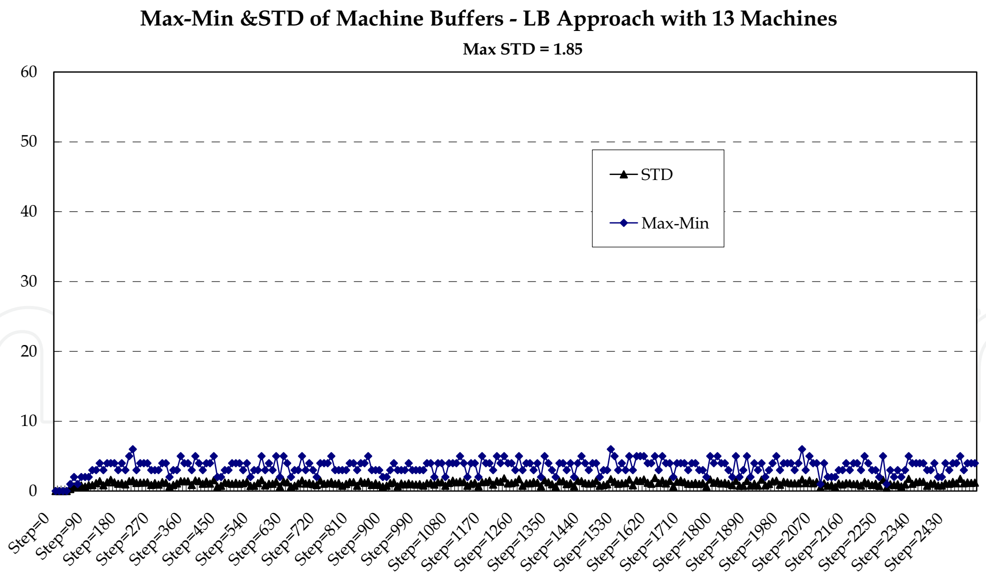
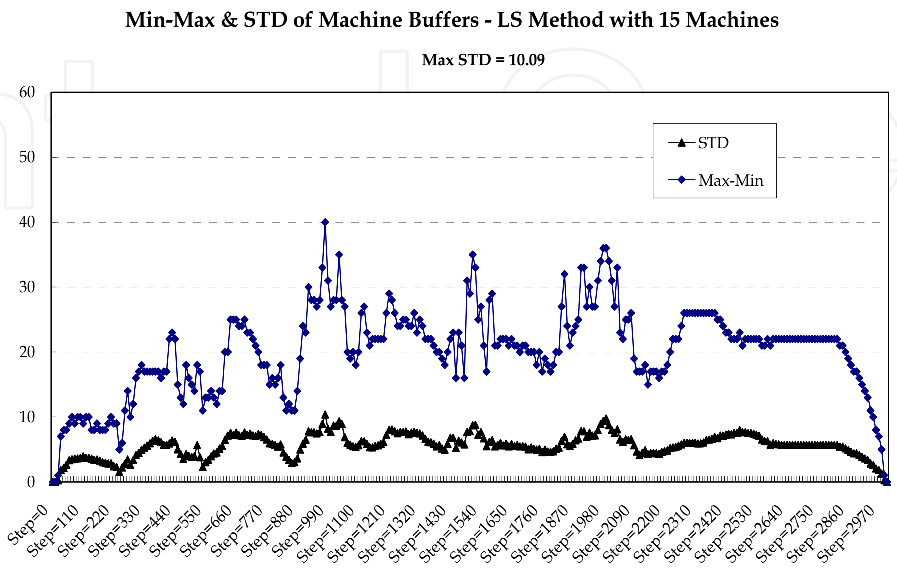


Figure 5(c). Thrashing phenomenon—13 machines



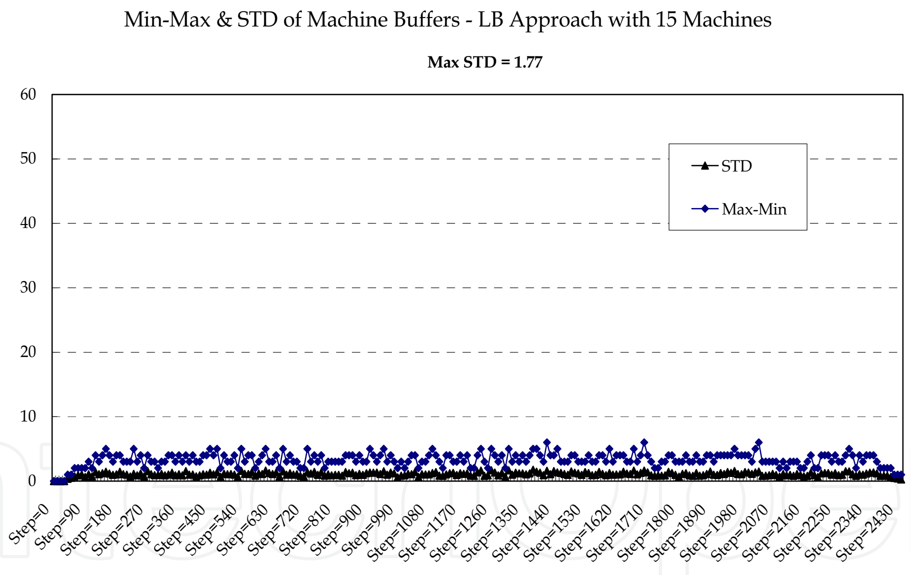


Figure 5(d). Thrashing phenomenon—15 machines

7. Acknowledgments

This research was partially supported by the U.S. National Science Foundation grant No. IIS-0326387 and U.S. AFOSR grant: FA9550-05-1-0454. This research was also supported in part by the Ministry of Economic Affairs under the grant 95-EC-17-A-02-S1-029 and the

National Science Council under the grants 95-2752-E-008-002-PAE and 95-2221-E-194-009. A. Shr is grateful to Ms. Erin Sinclair for English proofreading.

8. References

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

Arisha. A. & Young. P. (2004) Intelligent Simulation-based Lot Scheduling of Photolithography Toolsets in a Wafer Fabrication Facility. *2004 Winter Simulation Conference, Washington*, D.C., USA, 1935-1942.

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

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

Kumar. P. R. (1994) Scheduling Manufacturing Systems of Re-Entrant Lines. *Stochastic Modeling and Analysis of Manufacturing Systems*, 13, D.D. Yao (ed.), Springer-Verlag, New York, 87-110.

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

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

Mönch. L., Prause. M., & Schmalfuss. V. (2001) Simulation-Based Solution of Load-Balancing Problems in the Photolithography Area of a Semiconductor Wafer Fabrication Facility. *2001 Winter Simulation Conference*, Virginia, USA, 1170-1177.

Shen. Y. & Leachman. R. C. (2003) Stochastic Wafer Fabrication Scheduling. *IEEE Transactions on Semiconductor Manufacturing*, 1, 2-14.

Shr. A. M. D., Liu. A., & Chen. P. P. (2006a) A Load Balancing Scheduling Approach for Dedicated Machine Constraint. *Proceedings of the 8th International Conference on Enterprise Information Systems ICEIS 2006*, pp. 170-175, Paphos, Cyprus.

Shr. A. M. D., Liu. A., & Chen. P. P. (2006b) A Heuristic Load Balancing Scheduling Approach for Dedicated Machine Constraint. *Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems* (IEA/AIE'06), Lecture Notes in Artificial Intelligence (LNAI) 4031, M. Ali and R. Dapoigny (Eds), Springer-Verlag, Berlin Heidelberg, pp. 750-759.

Shr. A. M. D., Liu. A., & Chen. P. P. (2006c) A Load Balancing Method for Dedicated Photolithography Machine Constraint. *Proceedings of the 7th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services* (BASYS'06), IFIP International Federation for Information Processing, Vol. 220, Information Technology for Balanced Automation Systems, Shen, W. Ed., (Boston Springer), pp. 339-348.

Shr. A. M. D., Liu. A., & Chen. P. P. (2006d) Load Balancing among Photolithography Machines in Semiconductor Manufacturing. *Proceedings of the 3rd IEEE International Conference on Intelligent Systems (IEEE IS'06)*, pp. 320-325, London, England.

Vargas-Villamil. F. D., Rivera. D. E., & Kempf. K. G. (2003) A Hierarchical Approach to Production Control of Reentrant Semiconductor Manufacturing Lines. *IEEE Transactions on Control Systems Technology*, 11, 578-587.