Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

THE USE OF SLOW DOWN FACTORS FOR THE ANALYSIS AND DEVELOPMENT OF SCHEDULING ALGORITHMS FOR PARALLEL CLUSTER TOOLS

Robert Unbehaun

Institute of Applied Computer Science
Dresden University of Technology
Dresden, 01062, GERMANY.

Oliver Rose

Institute of Applied Computer Science
Dresden University of Technology
Dresden, 01062, GERMANY

ABSTRACT

In this paper, we describe the problem of developing scheduling algorithms for an environment of parallel cluster tools, which is a special case of the parallel unrelated machines problem. At first we will describe the problem under consideration in detail and then present our scheduling environment and the idea of using slow down factors to predict lot cycle times to evaluate schedules and parts of them. This article is more a conceptual kind of work containing mostly basic thoughts to illustrate facets of the problem and first solution ideas. Nonetheless the authors see a high potential in examining these questions. Little research has been done on that issue so far.

1 INTRODUCTION

Cluster tools are special integrated tools for wafer processing in semiconductor manufacturing. They are used to maximize throughput and reduce lot cycle times at the cost of a very complex behavior. Since wafers with different types of process steps can circulate in a cluster tool simultaneously it can be regarded as a job shop environment. Cluster tools work under vacuum conditions inside the tool which means very little particles that could possibly contaminate wafers. As a consequence, the clean-room quality outside the tool can be lower than in traditional fabs.

In the following three subsections we will first describe how cluster tools look like, what specifics in behavior there are and what makes scheduling of parallel cluster tools interesting. Section 2 gives a short literature review. Section 3 is concerned with the decomposition of the problem of scheduling parallel cluster tools as a whole and the description of the resulting subproblems. Section 4 will illustrate the first steps that have been taken to solve the problems described in 3 and address the use of slow down factors for lot cycle time prediction. In Section 5 we compare the two methods of cycle time prediction derived in Section 4 in terms of prediction errors and Section 6 will give a conclusion on the issue and a perspective for next research steps.

1.1 Components of Cluster Tools

The basic components of a cluster tool are:

- A vacuum mainframe with one or two wafer handling robots
- Several processing chambers, where some of them can be dedicated to identical processes and hence used in parallel
- Two load locks to pump to vacuum or vent to atmospheric conditions,
- Optionally there can be transfer chambers if there is more than one wafer handling robot
- An equipment front end module (EFEM) with an atmospheric wafer handling robot and several load ports, which is attached to the load locks (see Figure 1)

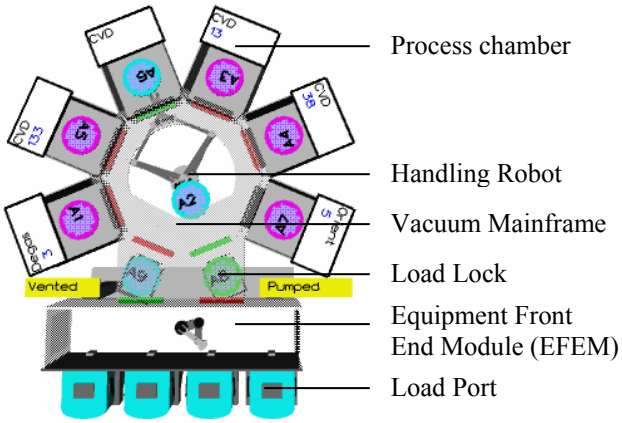

Figure 1: Example Cluster Tool

We assume that lots always contain 25 wafers, which all have the same process steps to be taken in the cluster tool. This sequence of process steps is usually referred to as "recipe". In the following examples lots are denoted with capital letters which also represent the recipe. If two lots have the same letter, all wafers of these lots follow the same recipe.

A typical product flow in a cluster tool starts with loading lots into one of the load ports. After that single wafers are consecutively transferred from the load port to the load lock by the atmospheric robot. Then the load lock will pump to achieve vacuum conditions. Now the main frame robot can transfer the wafer to its destination chamber where it is supposed to be processed. The next step depends on whether the wafer shall leave the system or is requested to be processed in another chamber according to its recipe. After the last process step the wafer will be guided through the load lock back to the load port.

Usually the main frame robot is a dualblade robot with the two blades either on the same side or opposite each other. Advantages compared to single blade robots are reduced wafer transfer times and with regard to multiple product flows a reduced amount of possible deadlocks as well.

1.2 Difficulties of Predicting Cluster Tool Behavior

Advantages and disadvantages of cluster tools were already discussed in detail in (Niedermayer and Rose 2003) and will be summarized briefly in the following. The biggest advantage of using cluster tools is the reduction of lot cycle times, since the processing of wafers is pipelined and thus the lot cycle time is only limited by the process time at the bottleneck chamber in the cluster tool rather than the sum of raw process times of all steps. Through the opportunity of processing lots of different recipes in parallel the throughput and the utilization are higher than they would be when processing every lot sequentially on a simple machine. Nonetheless, these advantages come at the cost of a very complex and difficult to predict behavior.

When processing a lot on a simple machine the cycle time is determined only by a constant or a single random variable. When processing a lot in parallel on a cluster tool it will have to share resources with wafers of other lots which leads to a cycle time strongly depending on the lot combination. In Figure 2 we illustrate how the cycle times and completion dates increase in parallel mode. The completion date of lot A and lot B, C_A and C_B, will be later than in single mode. But even if the cycle times of the lots are bigger ($\Delta C_A, \Delta C_B > 0$) the overall makespan $C_{\max}$ is reduced ($\Delta C_{\max} < 0$).

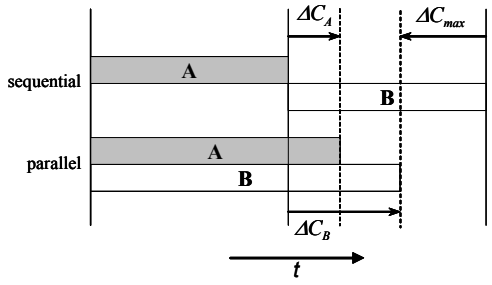

Figure 2: Comparison Single Mode – Parallel Mode

If we now consider that the cycle time slow down is different for each lot combination as well as for different lot sizes and even for different start delays between two lots, i.e., that for example lot B starts 5 minutes after the process start of lot A, the cycle times can only be determined by simulation, which is the only approach at the moment to evaluate cluster tool performance in detail.

1.3 Potential for Scheduling Analysis of Parallel Cluster Tools

Nowadays we can find a wide range of possible applications of cluster tools in semiconductor manufacturing. They are mainly used for etching, photolithography, chemical vapor deposition processes and also test processes.

Often these cluster tools are arranged together forming a tool group according to the type of process they are meant to handle. But nonetheless these tool groups can be very heterogeneous in terms of different tool properties like, e.g., the number and types of the process chambers, the wafer selection rule of the robot, the number of load ports and the type of the handling robot, which all affect cycle time, throughput and other common objective values. This leads to the necessity of a good scheduling policy for a list of lots either waiting to be processed on one of the tool group machines or being released from their preceding process steps at certain release dates.

Furthermore, interesting questions arise with the analysis of the tool group design, e.g., whether to take four cluster tools with four process chambers each or just two tools with eight chambers each or how much a heterogeneous tool group performs worse compared to a homogeneous one. Also, at which time a new lot of a certain type needs to be available for processing to assure the tool not to run under target utilization or even empty. One might ask how to distribute different process chambers over a number of cluster tools, too. Is it better to configure cluster tools such that only lots of a certain recipe can be processed or should a more flexible configuration be preferred? The list could be continued and there is little in literature to answer these questions so far.

2 RELATED WORK

A lot of cluster tool research is focused on issues of inside cluster tool scheduling and cluster tool controller dispatch rules as well as cluster tool simulation. Basic performance analysis and model development has been done in (Perkinson et al. 1994) and (Perkinson et al. 1996). Atherton (1995) gives a detailed introduction into cluster tools. Joo and Lee (1994) present a simulation framework with a virtual cluster tool controller included to reduce times for verfication of algorithms and behavior of real cluster tool controllers.

Niedermayer and Rose (2003) analyze cycle time delays occuring when lots of different recipes are processed in parallel in a cluster tool and present a method for lot cycle time prediction through slow down factors for cluster tools with two load locks.

LeBaron and Domaschke (2005) compare different dispatch heuristics for the cluster tool controller on differently configured cluster tool models using the commercial simulator ToolSim for evaluation.

So far no publications have been found adressing parallel cluster tool environments.

3 PROBLEM DESCRIPTION

The problem of scheduling parallel cluster tools can be derived from the parallel machine scheduling problem described in (Pinedo 2001). Since we also want to examine heterogeneous environments we obtain a problem of unrelated machines in parallel where a certain lot will have different process times on different cluster tools depending on its type.

Since this is more a conceptual kind of work we do not define a concrete objective function but want to give a first understanding of the problem. The technique of using slow down factors for cycle time prediction presented later is a first approach for analyzing cycle times and cycle time related objectives like, e.g., throughput.

For our case we can decompose the problem into two subproblems, which is at first the assignment of the lots to one of the cluster tools and second the sequencing of the queue for each tool with regard to the fact that at each cluster tool a number of lots equal to the number of given load ports can be processed simultaneously (parallel mode). This depends on the lot processing strategy used in the cluster tool, since it can also be configured to process the lots in the load ports sequentially (single mode) or use some of the load ports for lot buffering (lot buffering mode). The buffering is very common in reality, to avoid utilization decrease through missing lots, which is caused by transport delays during lot delivery from the preceding tool or stocker to the cluster tool. Single and parallel mode were already discussed in (Niedermayer and Rose 2003).

In the next two subsections we will describe the additional constraints and parameters, which need to be considered when solving the partitioning and sequencing problem for cluster tools rather than simple machines in parallel.

3.1 Partitioning Problem

With a list of lots given to be scheduled, we have to decide on which machine to put each single lot. With cluster tools there are some additional variables and constraints which should be taken into account to achieve a good solution. If we assume the cluster tools are different they can vary in the following scheduling relevant parameters:

Number of process chambers: How many process chambers are available to process each step of a certain recipe? How many of them are used by lots of other recipes as well? Both will affect lot cycle time if lots are processed in parallel.

Number of load ports: How many lots can be processed at the same time at maximum?

Lot processing strategy:

- Parallel mode – Wafers of all available lots are circulating in the cluster tool at the same time. All load ports are served.
- Single mode – All lots are processed one after another.
- Lot buffering mode – Only lots of different recipes are processed in parallel. Lots with identical recipes are processed sequentially. Hence, not all load ports are served at the same time.

Type of the wafer handling robot:

- Single blade – The robot can only transport one wafer at a time.
- Dual blade – The robot can transport two wafers at a time. The blades are either opposite each other or on the same side.

Wafer selection rule:

- Pull – A wafer will only be picked by the robot if the destination chamber is available. If more than one wafer is waiting, the one which was waiting the longest will be picked.
- Push – This rule is meant for dual blade robots only. The robot picks a wafer and transports it to the destination chamber. If this is busy, the robot will pre-position in front of it and wait to swap processed and unprocessed wafer.
- *Others Rules* – There exist many other dispatch rules. Besides push and pull (LeBaron and Domaschke 2005) present two more priority based rules.

An important constraint is that not all cluster tools might be able to process all types of recipes.

Besides these decision restrictions there are other optional decisions which can be made. E.g., we can choose whether we want to assign mostly lots of the same recipe or mostly lots of different recipes to a cluster tool. The first would result in smaller cycle times but lower utilization, while we expect to obtain higher utilization and larger cycle times in the second case. Other decisions are also possible.

3.2 Sequencing Problem

For the lot sequencing in the queue of each cluster tool we have to take care of certain constraints.

The most important problems we obtain are the interferences occurring when processing different types of lots in parallel on the same cluster tool. These interferences cause an increase of the lot cycle times which might be smaller or larger mainly depending on the actual lot combination used.

On the other hand it is no good idea to always just process only lots of one recipe in parallel since the wafers of these lots usually occupy only a certain number of the given process chambers according to their recipe. Some process chambers will not be used then and thus the utilization of the tool will not be sufficient with respect to the fact that cluster tools are the most expensive equipment in semiconductor manufacturing.

Even if the cluster tools would be configured such that lots of different recipes have to be processed separately this is not a good idea, since in case that one cluster tool fails a certain type of recipe could not be handled anymore. The tool group is not very tolerant with breakdowns, then.

Thus, it is important for lot sequencing to order the lots in a sequence which minimizes the time delays caused through interferences but assures a high utilization of the tool at the same time.

4 SOLUTION APPROACH

In the following we will describe the steps taken so far to become familiar with the problem.

To develop algorithms for the solution of the problem described, it is at first necessary to develop a framework which is appropriate to model and analyze parallel machine environments in general and parallel cluster tool environments in particular. This includes the integration of a simulator, which is applicable to represent the wide range of possible tool configurations used in reality as well as to handle the complex wafer routing and the behavior of real cluster tools as a whole. Otherwise we would not have any indication of cluster tool behavior in reality since there are no mathematical models for prediction available.

We made a prototype implementation of a java-based scheduling and simulation framework which is appropriate to read all scheduling relevant input data concerning cluster tool configuration, lot data and wafer recipes via XML-files and to produce text based output data. More over, any user defined simulator for schedule evaluation can be addressed via a corresponding interface. We designed our data structure in a way that allows an easy integration of additional scheduling algorithms to speed up the implementation and testing of newly derived heuristics. All this makes the framework a suitable tool to take investigations in the field of scheduling parallel cluster tools.

For simulation we use ToolSim from Brooks Automation, Inc., which is already established and used by the industry for several years now. With this simulator we are able to model most up to date cluster tool equipments in a sufficiently detailed manner.

4.1 Slow Down Factor Definition

Since it is too time consuming to make a large amount of simulation runs with the ToolSim simulator, e.g. to apply a local search algorithm, we need a method to estimate lot cycle times as accurately as possible for a single cluster tool. We have chosen to use the idea of the slow down factor analysis for this prediction, because it is a fast technique to evaluate whole schedules with regard to lot cycle times and lot completion dates. It is already known in the literature (Niedermayer and Rose 2003) but only for the cluster tool configurations with just two load locks and no EFEM with a varying number of load ports. Hence, we need to generalize the problem first. After that, it is possible to use the method independently of the cluster tool configuration.

As illustrated in Figure 2 lots processed in parallel slow down each other in processing.

Definition 1 *The slow down factor of lot A while processed in parallel with lot B is defined as*

$$SDF(A, A+B) = \frac{CT(A, A+B)}{CT(A)}, \qquad (1)$$

where $CT(A, A+B)$ *is the cycle time of lot A when it is processed in parallel with lot B and* $CT(A)$ *represents the cycle time of lot A when it is processed alone. It always holds that* $SDF \geq 1$ (Niedermayer and Rose 2003).

Thus, if there are more than two load ports available at the cluster tool we can also have slow down factors like $SDF(A, A+B+C+D)$ or $SDF(A, A+B+A)$. With the last example there is a problem occurring in terms of whether the slow down factors of both of the lots of type A are the same or not. In our experiments which ran under lot buffering mode, since ToolSim does not provide the option of real parallel processing of all lots independent of their type, we discovered that the slow down factors of the second lot is always larger than the first.

Table 1: Example Slow Down Factors

SDF	Value
(A(1),A(1))	1.0000
(A(1),A(1)+A(2))	1.0262
(A(2),A(1)+A(2))	1.9075
(A(1),A(1)+A(2)+A(3))	1.0262
(A(2),A(1)+A(2)+A(3))	1.9343
(A(3),A(1)+A(2)+A(3))	2.8114
(B(1),B(1))	1.0000
(B(1),B(1)+B(2))	1.0137
(B(2),B(1)+B(2))	2.0040
(B(1),B(1)+B(2)+B(3))	1.0137
(B(2),B(1)+B(2)+B(3))	2.0177
(B(3),B(1)+B(2)+B(3))	3.0080

For the experiments in Table 1 we used a cluster tool configuration with four process chambers, three of them in parallel. We have two recipes, which both have only one process step in the cluster tool. Lots of recipe *A* can be processed in one of the parallel chambers and lots of recipe *B* only in the fourth one. The numbers in brackets represent the load port the lots are assigned to.

For both recipes we can see that the second and the third lot nearly need twice and three times as much time as the first lot, which proves that the cluster tool processes lots of the same recipe sequentially starting at the first load port. If there are parallel chambers available like with recipe *A*, cycle times of second and third lots are reduced since wafers of the second lot (and third respectively) can already be introduced into the system as soon as one of the parallel chambers is not needed anymore by a wafer of the preceding lot. In this case the slow down factor of the first lot is greater than one unlike the case of alone processing. It can be explained with the shared use of the transport robots which also start transporting wafers of the second lot when the first lot is nearly finished.

4.2 Slow Down Factor Calculation

Now, to achieve a possibility for schedule cycle time prediction independently of the lot processing mode, we need to simulate the load port assignment combinations for all given recipes. This only needs to be done once and is then meant to be used for quick schedule evaluation in the optimizing algorithms.

Since raw process times for lots simulated in single mode are different we need to assure that during the simulation of a recipe combination lots of all recipes are available at the load ports in the same combination unless the lot with the longest raw process time is finished, to assure parallel processing during the whole time (see Figure 3). Hence, we need to include an additional queue of lots which are sorted such that always a lot with the same recipe like the one that is leaving the system can be assigned to the free load port.

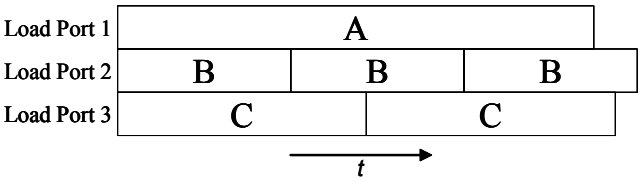

Figure 3: Load Port Assignment over Time

Difficulties arise with the estimate of times when a lot is finished and the corresponding order of the queue. Our first approach is to relate the raw process times for every recipe to each other and put lots into the queue with respect to these relative values.

Example 1:

Three recipes are given and we have the simulated raw process times (single mode) shown in Table 2.

Table 2: Example Lot Process Times

Recipe	Raw process time RPT_i
A	10
B	20
C	40

Now, the relations are $C = 2 \cdot B = 4 \cdot A$. There are three load ports given and we want to obtain all slow down factors for the load port assignment ($C + A + B$). We have to put lots of recipes *A* and *B* into the queue to guarantee the lot of recipe *C* not to run alone in the cluster tool after *A* and *B* were finished. We need at least two lots of *B* and for every lot of *B* two lots of *A* to be processed. Hence, our additional queue will be (*A*, *B*, *A*, *A*). The lots of *A* and *B* initially assigned to the load ports are also counted.

At this point, the problems with that method are already conceivable, since it relies that the process time relations based on the raw process times also hold for the simulation of a recipe combination with corresponding interferences. This method works fine as long as there are only lots of two different recipes assigned to the load ports since then the queue only includes lots of one recipe. As soon as there are lots of at least three different recipes assigned to load ports there is no guarantee any more that a leaving lot of a certain recipe is replaced with a lot of the same recipe all the time. At least it can be guaranteed that there will be wafers of additional lots circulating in the cluster tool until the recipe with the longest raw process time is finished and thus there is a shared use of the transport robots and load locks which will result in an almost accurate slow down factor.

4.3 Cycle Time Prediction Using Slow Down Factors

We now want to predict the simulated cycle times by using the calculated slow down factors and we divide the predic-

tion into prediction of lot cycle times and prediction of simulated lot completion dates where a lot is finished processing. So far we derived two different algorithms for prediction.

Method 1:

If we have given our queue of a certain number of lots, we start assigning them iteratively to the load ports until no load port is available any more. For these first lots we calculate the cycle time CT_i of each lot i through $CT_i = RPT_i \cdot SDF_i$, where RPT_i represents the raw process time of lot i processed in single mode and SDF_i the slow down factor for lot i in the corresponding recipe combination which is currently given at the load ports. For these lots cycle times and completion dates are equal since simulation starts at time $t = 0$. After that we take the lot which is finished first and replace it with the next lot from the queue. For this lot the start time equals the completion date of the preceding lot.

Now there is a problem if we want to calculate cycle time and completion date. We cannot just take the slow down factor of the new load port recipe combination that is the result of the exchange. E.g., if the lot exchange occurred at the first load port and we would take the corresponding recipe combination where the new lot is at the first position we obtain a wrong slow down factor since in the slow down factor calculation the simulator always starts to process the lot in increasing load port order. Hence, we have to take the combination where the new lot is at the last position since previously assigned lots are processed first if there is no complete parallel processing. Example 2 illustrates the problem.

Example 2:

With respect to Table 1 our current load port recipe combination is (*A*(1) + *A*(2) + *A*(3)). The lot of recipe *A* at load port 1 will be replaced by another lot of recipe *A*. Since our calculation needs to be independent of the lot processing mode (parallel or lot buffering) we cannot take *SDF*(*A*(1),*A*(1)+*A*(2)+*A*(3)) but *SDF*(*A*(3),*A*(1)+*A*(2)+*A*(3)) instead, because this case represents newly assigned lots which may have to wait until previous lots are finished depending on the processing mode. If we have a raw process time of $RPT_4 = 10$ given the correct cycle time prediction for the new lot (the fourth from the queue) would be $CT_4 = 28.11$ rather than $CT_4 = 10.26$.

The completion date of the new lot is then calculated as the sum of start time and cycle time. After that the lot with the next earliest completion date will be replaced with the next lot from the queue and so on.

Method 2:

Similar to Method 1 we start to calculate cycle times and completion dates for the initial load port assignment and always replace the lot with the next earliest completion date successively. The slow down factor will also be chosen from the load port recipe combination where the recipe of the new lot is at the last position. The difference is now that, if we replace a lot with a resulting change of the load port recipe combination, the completion dates of all assigned lots will be adjusted according to the new combination. The idea is that with a new lot introduced the extend of cycle time slow down will change for all lots and needs to be taken into account not only for the new lot. The cycle time for the new lot is calculated like in Method 1 and the completion date is $CD = t + CT$ where t is the actual time a new lot is introduced. For a lot i that is already processing we calculate the new completion date $CD_{i,new}$ through

$$CD_{i,new} = t + \frac{CD_{i,old} - t}{SDF_{i,old}} \cdot SDF_{i,new}, \qquad (2)$$

with $CD_{i,old}$ as the former completion date used so far, $SDF_{i,old}$ as the former slow down factor and $SDF_{i,new}$ as the slow down factor for the new load port recipe combination.

5 EXPERIMENTAL RESULTS

In the following we present some short experiments to compare the two suggested methods in terms of prediction errors in relation to the simulated values. Table 3 contains the experimental design chosen. We created a random queue of either 10 or 40 lots, where one of either two or three recipes was assigned to each lot by chance. The queue was then processed on a cluster tool either configured with two or three load ports. The cluster tool always has four chambers. For runs with two recipes there are three of them in parallel for recipe *A* and one for *B*. For three recipes only two are in parallel for *A* and one chamber is dedicated to each of *B* and *C*. For each design we made ten independent runs. The given process times per wafer may not be mistaken for the simulated raw process times of a lot with a certain recipe (*RPT*). The given times just represent the duration of the concrete wafer processing procedure in a certain process chamber. Only through simulation we can obtain the minimal time a whole lot would need to be processed at the cluster tool.

Table 3: Experimental Design

Variable	Values
Number of Load Ports *LP*	2, 3
Number of Recipes *R*	2, 3
Process Time per Wafer of Recipe *A* [sec]	60
Process Time per Wafer of Recipe *B* [sec]	110
Process Time per Wafer of Recipe *C* [sec]	210
Number of random lots *L*	10, 40
Random Iterations	10
Number of Runs	80

Table 4: Prediction Errors (CT - Cycle Time, CD - Completion Date)

			Method 1		Method 2	
L	*R*	*LP*	∅ CT Error	∅ CD Error	∅ CT Error	∅ CD Error
10	2	2	23.6%	10.8%	23.0%	19.7%
		3	17.0%	8.0%	24.0%	20.3%
	3	2	17.7%	7.4%	15.3%	9.8%
		3	14.7%	7.7%	16.1%	11.9%
40	2	2	23.6%	7.6%	24.9%	23.0%
		3	20.0%	7.1%	31.5%	32.0%
	3	2	17.8%	6.5%	16.8%	15.7%
		3	20.9%	6.7%	24.9%	22.4%

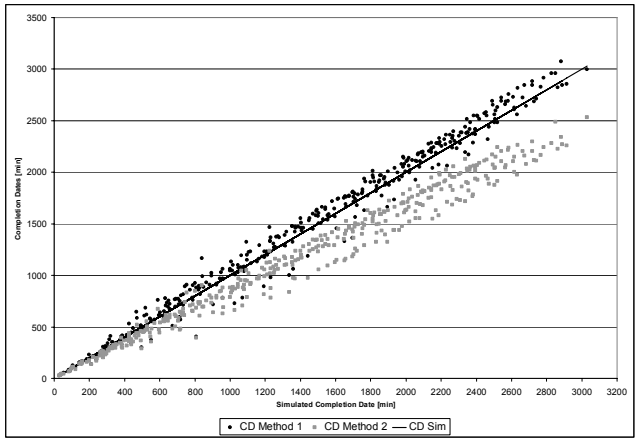

Figure 4: Calculated Completion Dates for L=40, R=3, and LP=2

We conclude from Table 4 as well as Figures 4 and 5 that Method 1 outperforms Method 2 especially with regard to the simulated completion dates, where Method 1 is usually deviating around an average of 7% while Method 2 has average deviations of up to 32%. The average cycle time deviation seems to be nearly the same for both methods.

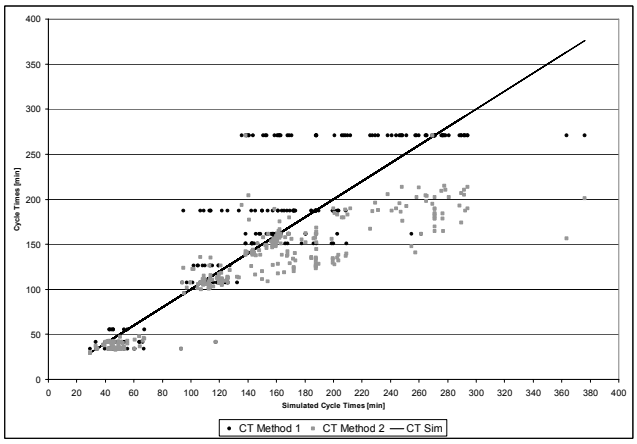

Figure 5: Calculated Cycle Times for L=40, R=3, and LP=2

The better performance of Method 1 in terms of completion date deviation results from the fact that in many cases positive deviations compensate negative ones, while Method 2 mostly calculates completion dates which are smaller than the simulated values and hence no compensation can take place. The same holds for the cycle time deviation but since with Method 1 positive deviations outweigh the negative ones, there is less compensation, which leads to as bad deviations as Method 2.

In Figure 5 we can see the values of Method 1 grouped on some kind of levels. Obviously, the amount of possible cycle time values is very limited. The reason is that only the raw process times and the single amount of available slow down factors are used for calculation.

We further derive from Figures 4 and 5 that the slow down factor adjustment used in Method 2 results in a reduction of cycle time which is too strong to give a good prediction. It needs further investigation to determine whether a more sensitive adjustment is applicable or not.

Finally we can say that so far Method 1 should be preferred especially for predicting lot completion dates.

6 CONCLUSION

In this paper, we presented our first steps on our way to develop scheduling heuristics for parallel cluster tools. We provide an idea of the difficulties occurring with these machine environments and described the necessity of a cycle time prediction method applicable for varying cluster tool configurations since otherwise single simulation runs take very long. The slow down factor analysis was our first approach and there need more experiments to be done to see whether this technique is applicable in terms of prediction quality and to guarantee suitability for schedule evaluation used in future optimizing algorithms. We compared two methods for cycle time as well as completion date prediction of simulated values based on slow down factors and found Method 1 especially worth for evaluation of lot

completion dates so far. Further research has to be done to show that the methods are applicable independent of the cluster tool configuration and to reduce prediction errors.

REFERENCES

Atherton, L.F. and R.W. Atherton. 1995. *Wafer Fabrication: Factory Performance and Analysis*. Kluwer.

Joo, Y.-J. and T.-E. Lee; 2004. "Virtual Control: A Virtual Cluster Tool for Testing and Verifying a Cluster Tool Controller and a Scheduler". *IEEE Robotics & Automation Magazine*, 11(3), 33-49.

LeBaron, T. and J. Domaschke. 2005. "Optimizing Robot Algorithms with Simulation". In *Proceedings of the 2005 Winter Simulation Conference*. 2211-2217.

Niedermayer, H. and O. Rose. 2003. "A Simulation-based Analysis of the Cycle Time of Cluster Tools in Semiconductor Manufacturing". In *Proceedings of the 15th European Simulation Symposium*.

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

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

Pinedo, M. 2001. *Scheduling. Theory, Algorithms, and Systems*. 2nd edition. Prentice-Hall.

AUTHOR BIOGRAPHIES

ROBERT UNBEHAUN is a PhD student and member of the scientific staff of Prof. Dr. Oliver Rose at the Chair of Modeling and Simulation of the Dresden University of Technology, Germany. He received a M.S. degree in computer science from Ilmenau University of Technology, Germany. His research interests include optimization approaches for complex manufacturing environments through development and analysis of scheduling algorithms for material flow control. His e-mail address is <robert.unbehaun@inf.tu-dresden.de>.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Institute of Applied Computer Science of the Dresden University of Technology, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI. His web address is <http://www.iai.inf.tu-dresden.de/ms>.