FabTime Cycle Time Management Newsletter

Volume 9, No. 3 March/April 2008

Information

Mission: To discuss issues relating to proactive wafer fab cycle time management

Publisher: FabTime Inc. FabTime sells cycle time management software for wafer fab managers. New features in this month include enhancements to the goal-setting user interface (for system administrator and personal goals), and support for site-specific WIP attributes and filters.

Editor: Jennifer Robinson

Table of Contents

Welcome

Welcome to Volume 9, Number 3 of the FabTime Cycle Time Management Newsletter! We've been quite busy at FabTime in March, with multiple cycle time management courses, and the commencement of two new FabTime installations. [I also moved - if you need my new direct phone number, just email Jennifer.Robinson@FabTime.com]. I hope that business is strong for you all. In this issue, we have community announcements about the second issue of Fab Engineering and Operations Magazine and a milestone reached by the Fab Owners Association. Our software user tip of the month describes how to use FabTime's new Queue Limit Lot List chart, which shows the non-held lots in queue that have exceeded, or are in danger of exceeding, a user-specified threshold.

We have one subscriber discussion question this month, about batch loading rules. In responding to this question, we realized that it has been more than five years since we last discussed batching in detail in the newsletter. Therefore, we decided to discuss batching in our main article this month. Specifically, we review the cycle time benefits of a greedy vs. a full batch policy, with examples, and also provide a simple rule of thumb for using look-ahead information in the batch formation decision. We welcome subscriber feedback, especially about experiences with greedy vs. full batch policies and incorporating look-ahead information into the batch loading decision.

Thanks for reading!—Jennifer

FabTime

Tel: (408) 549-9932
Fax: (408) 549-9941
www.FabTime.com
Sales@FabTime.com

Community News/Announcements

New Issue of FEO Magazine Available

The second issue of Fab Engineering and Operations Magazine was published at the end of February. FEO Magazine is a new industry publication, available as a free PDF download. It focuses on the day to day operational issues faced by existing fabs. This makes FEO Magazine of direct interest to most FabTime cycle time management newsletter subscribers. Several members of the FEO Magazine editorial board are also long-time FabTime newsletter subscribers. If you haven't already seen it, I encourage you to download the new issue from http://www.feomag.com.

Fab Owners Association (FOA) Reaches Membership Milestone

Cupertino, Calif. – February 26, 2008 – The Fab Owners Association (FOA), the association of semiconductor / MEMS manufacturing executives and suppliers, has announced the addition of it newest device-maker member, Diodes Incorporated (NASDAQ:DIOD), www.diodes.com. Diodes is a leading global manufacturer and supplier of high-quality application-specific standard products within the broad discrete and analog semiconductor markets.

"We are excited that Diodes has chosen to join the Fab Owners Association, bringing our membership to 49 members," stated L.T. Guttadauro, executive director of the FOA. "With the addition of three new associate members, Cypress Systems, www.cypress.com/systems/, Matheson Tri-Gas, www.mathesontrigas.com/ and Semplastics, www.semplastics.com/ the FOA now stands at 52 members."

"Our mantra of seeking and sharing semiconductor manufacturing efficiencies through collaboration continues to pay dividends for our membership," said Mr. Guttadauro. "Device manufacturers are attracted by our level playing field, manufacturing benchmark studies and our aggregate purchasing programs. Our associate member companies are attracted by the growing marketing opportunities presented by our membership."

The FOA's device maker members are the following companies: AMI Semiconductor, Austriamicrosystems AG, Avago Technologies, Cypress Semiconductor, Delphi Microelectronics Center, Diodes, Inc., Fairchild Semiconductor, Freescale Semiconductor, International Rectifier, Intersil, Jazz Semiconductor, MagnaChip Semiconductor, Micrel Semiconductor, Microchip Technology, ON Semiconductor, Skyworks Solutions, and X-FAB.

FOA device maker member companies represent approximately 1,000,000 eight-inch- equivalent monthly wafer starts and US$19 Billion in annual revenue. FabTime is an associate member of the FOA.

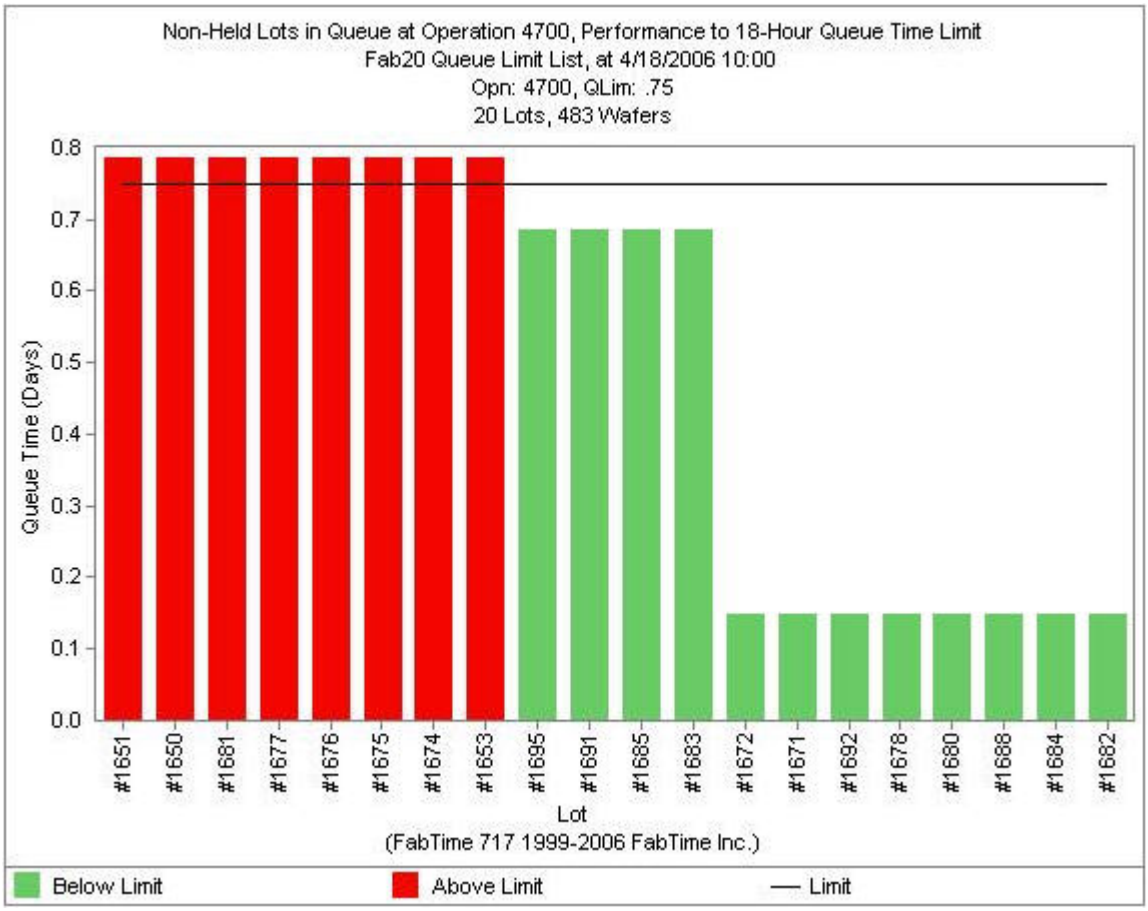
FabTime welcomes the opportunity to publish community announcements. Send them to newsletter@FabTime.com.

FabTime User Tip of the Month

View Lot Performance to a Queue Time Limit

Do you need to know which lots have been in queue for more than some specified time at a particular tool or operation? Would you also like to be able see lots approaching the queue time limit? The queue time limit could be a hard limit, based on time constraints between process steps, and indicating that lots need to be sent back for reprocessing. Or it could be a target, based on your desired turns rate for the tool or operation. In either case, FabTime's new Queue Limit Lot List chart can show you what you need. The Queue Limit Lot List chart shows, according to whatever filters you have specified, the list of all non-held lots currently in queue, color-coded red or green depending on if each lot has exceeded the user-specified queue time limit.

To use this chart, select Queue Limit Lot List from the WIP Charts category on the FabTime Charts page. Enter your filters of interest (e.g. filter for a particular area, tool group, or operation). Also enter your target queue time limit in the "QLim:" filter (located near the bottom of the main set of filters to the left of the chart, between the "Age" and "SQL" filters). The units for the queue time limit should match the value you have selected in the "U/M" dropdown, immediately below the SQL filter. The default unit for this field is days. Once you enter a QLim value, FabTime will display the list of all non-held lots currently in queue, with a line across the chart showing the queue time limit. Bars will be colored red for each lot that has exceeded the queue time limit, and green otherwise. An example, for an operation



with a target queue time of no more than 0.75 days, is shown below:

Note that the list of lots that have exceeded some target queue time limit can also be obtained on the WIP Lot List chart, by setting the "Hold" drop-down list to "Non-Hold", the "Que" drop-down to "In Queue", and the "Age>=" field to your target queue time limit. However, in this case FabTime will only display the lots that have exceeded the limit. What the Queue Limit Lot List chart adds is the ability to see, in red, the lots that have exceeded the limit, while also seeing the lots that have not reached the limit. The WIP Lot List for the previous example, if filtered to show lots with Age >= .75 days, would only show the eight red lots. In this case, we would not be able to see the next four lots, which are approaching the queue time limit, or to see the other, newer lots.

We hope that you'll find this new chart useful. If you have any questions about this feature (or any other software-related issues), just use the Feedback form in the software.

Subscriber Discussion Forum

Batch Loading and Look-Ahead Policies

An anonymous subscriber wrote: "I am currently looking for any information you have on batch loading, including look-ahead information. I read the article posted in the library on your website. I am currently looking at improving our procedures, however, would like more information and especially data.

Also, what do you use to simulate different scenarios? What software etc…"

FabTime Response: We have written about batching, including look-ahead, in FabTime newsletters 2.1 and 3.8. We also have some additional detail as part of our cycle time management class, though that information isn't generally available outside of the course. We don't have any hard data about people's success with particular loading policies, including look-ahead, in their fabs, and we would like to open up this topic to anyone who might have something to share. We have also been spurred by this comment to discuss batch loading issues in the main article for this issue. Further details are below.

Regarding simulation, we use a product called Factory Explorer, sold by Wright Williams and Kelly, for fab simulation. FabTime's Frank Chance originally developed that product, and sold it to WWK (www.wwk.com) about 10 years ago. While we think that it's an excellent product, we are somewhat biased because of our background with it. For small simulation models with animation, to understand how a particular system works, and for illustration in our training class, we use ProModel.

FabTime welcomes the opportunity to publish subscriber discussion questions and responses. Send your questions to Jennifer.Robinson@FabTime.com.

Batch Loading Policies for Wafer Fabs

Introduction

We have written about batch loading policies before in the FabTime newsletter (Issues 2.01 and 3.08). However, our last article on this topic was published in 2002. We thought, particularly in light of the subscriber question above, that this would be a good time for a refresher.

Background: Greedy vs. full batch Policies

The term "batch loading policies" refers to decisions made for the loading of tools that can process more than one lot at one time. Typically, the process time is independent of the number of lots included in each batch (the process time may be recipe-dependent, but in this case only lots with the same processing time are included in the same batch). The classic batch tool is a furnace that can process anywhere from one to eight 25-wafer lots, and has a relatively long process time, typically eight hours or more. When there is a full load available to run in the furnace, the situation is straightforward. Processing should begin on the tool as soon as possible. However, the situation where there are only partial batches available is more controversial. In this case, a decision must be made on whether to start a non-full batch run, or wait until there are sufficient lots to form a full batch. Reasons for waiting include:

- Desire to minimize the number of runs on the tool to reduce the need for run-based PMs.
- Desire to minimize the number of runs on the tool to reduce consumables use.
- Desire to wait for immediately upcoming lots, so that they are not required to wait during a long process time.
- Desire to run the tool more efficiently, without "wasting" the empty spots, since this unused capacity can never be restored.

These reasons sometimes drive fabs to implement what's called a full batch policy - only run the tool when there is a full, or nearly-full, load available. And a full batch policy certainly makes sense in situations of high cost consumables, or very expensive PMs. The problem with a full batch policy, however, is that it can significantly increase cycle times, both at the batch tool and throughout the fab. If you impose a full batch policy on a tool that is not very heavily utilized, what happens is that the first lots to arrive at the tool wait quite a long time for a full batch to become available, accruing considerable cycle time. Then, after the full batch is finally run, the batch is sent downstream, creating a highly variable arrival process to the next step. If all of the lots in the batch are going to the same per-lot or per-wafer tool group, then most of the lots will have to wait again, accruing more cycle time. In general, running large batches introduces variability into the fab, and drives up cycle time.

By contrast, a greedy policy says "if the tool is available and there is WIP at the tool, go ahead and start processing, even if you don't have a full batch". A greedy policy results in much lower cycle times for tools that aren't heavily utilized. Lots that arrive to the tool get processed more quickly, instead of having to wait for a full batch. Then smaller batches are sent downstream, resulting in less variability, and hence lower cycle times downstream.

Here's the important thing about running a greedy policy. While it generates much lower cycle times for lightly utilized tools, even if a tool is heavily utilized, the greedy policy often gives good performance, approximately as good as a full batch policy at very high utilizations. The reason for this is that the greedy policy only says "go ahead and run, even if you don't have very many lots available." It doesn't say "run small loads when there is a lot of WIP

there." If you implement a greedy policy on a heavily utilized tool, usually there will be a full or near full batch waiting to be processed whenever the tool becomes available, and that full batch will be run. Even if you occasionally run a batch that only has one or two lots in it, by the time that batch finishes, a full load will likely be waiting for the next run. As a result, the greedy policy is what we call robust in the presence of utilization changes. It works well whether the utilization is high, medium, or low. By contrast, a full batch policy only works well when the utilization is high, and generates a significant cycle time penalty when the utilization is lower. If you have a large variety of furnace recipes, you may need to use a near-greedy policy that requires only a few lots (but more than one) to be ready in order to start processing – a strict greedy policy may misbehave in this case.

The reason that we want to have a robust batch loading policy, of course, is that product mix changes frequently in fabs. A tool that is heavily utilized one month might drop off in the next month. If you have a full batch policy in place on a tool that's quite busy, and then the loading drops off, and you don't change the loading policy, you can end up incurring extra cycle time. If you have a greedy policy, or near-greedy policy, in place, the tool will continue to perform well. Examples are shown below.

Examples: Greedy vs. Full Batch Policy

Here we illustrate this difference between running greedy vs. full batch policies using numeric examples. These examples were also published in Issue 2.1.

A. Single tool, constant time between arrivals

Full Batch Policy

Suppose we have a single furnace that can process up to eight lots at one time, and has an eight-hour process time (constant). If a single lot arrives every two hours (constant time between lot arrivals), then on average the furnace will be loaded to 50% of its capacity (since it can process eight lots every eight hours, but only four lots arrive every eight hours). (We're neglecting downtime for this example). Suppose that the furnace has just started processing a batch, and call this time zero. Let's look at what happens when we wait to form full batches. We won't be able to start another batch until eight lots have arrived, 16 hours from now. We get the following pattern of arrival, start process, and queue times, where queue time is simply Start Process Time - Arrival Time, where AT = Arrival Time, ST = Start Time, and QT = Queue Time:

Lot #: 1. AT: 2. ST: 16. QT: 16-2=14
Lot #: 2. AT: 4. ST: 16. QT: 16-4=12
Lot #: 3. AT: 6. ST: 16. QT: 16-6=10
Lot #: 4. AT: 8. ST: 16. QT: 16-8=8
Lot #: 5. AT: 10. ST: 16. QT: 16-10=6
Lot #: 6. AT: 12. ST: 16. QT: 16-12=4
Lot #: 7. AT: 14. ST: 16. QT: 16-14=2
Lot #: 8. AT: 16. ST: 16. QT: 16-16=0

The average queue time is (14+12+10+8+6+4+2)/8 = 56/8 = 7 hours. Because everything is constant in this example, the entire pattern just repeats, and thus the average queue time across all lots for the full batch scenario is eight hours, nearly equal to the raw process time of eight hours.

Greedy Policy

Now suppose that instead of forming full batches, we use a greedy policy and start processing a batch whenever the furnace is free, and lots are available. In this case, we'll start a new batch every eight hours (every time the furnace becomes free). Starting with the same starting point as previously, we have the following pattern, where AT = Arrival Time, ST = Start Time, and QT = Queue Time:

Lot #: 1. AT: 2. ST: 8. QT: 8-2=6
Lot #: 2. AT: 4. ST: 8. QT: 8-4=4
Lot #: 3. AT: 6. ST: 8. QT: 8-6=2
Lot #: 4. AT: 8. ST: 8. QT: 8-8=0

Lot #: 5. AT: 10. ST: 16. QT: 16-10=6
Lot #: 6. AT: 12. ST: 16. QT: 16-12=4
Lot #: 7. AT: 14. ST: 16. QT: 16-14=2
Lot #: 8. AT: 16. ST: 16. QT: 16-16=0

Now the average queue time is (6+4+2+0+6+4+2+0)/8 = 24/8 = 3 hours. We've eliminated four hours of queue time, on average, for all lots, by not forcing a low-utilization furnace to be loaded to full all the time.

B. Single-Tool Simulation Results

Obviously, the above example is unrealistic - with constant process and interarrival times. We ran a series of simulation models of this system, with highly variable times between arrivals (exponential). We varied the system loading, to look at the interaction between furnace utilization and minimum batch size, and also varied the threshold value. Here are the results for minimum batch sizes of 1 (greedy) and 7 (nearly full). We ran each simulation for 5 years, and only include results from a single replication for illustration. The numbers displayed are x-factor (cycle time divided by raw process time). So, the number 2 indicates that the total cycle time was twice the eight-hour process time, or 16 hours total.

Util: 20%. Greedy: 1.45. Near-Full: 2.87
Util: 30%. Greedy: 1.48. Near-Full: 2.24
Util: 40%. Greedy: 1.50. Near-Full: 1.93
Util: 50%. Greedy: 1.51. Near-Full: 1.77
Util: 60%. Greedy: 1.53. Near-Full: 1.66
Util: 70%. Greedy: 1.57. Near-Full: 1.63
Util: 80%. Greedy: 1.65. Near-Full: 1.66
Util: 90%. Greedy: 2.00. Near-Full: 2.00
Util: 95%. Greedy: 2.47. Near-Full: 2.45

Here we see that until the furnace is loaded to about 90%, a greedy (minimum batch size of one) policy results in lower cycle times than a full batch policy. At high utilizations there is a very slight improvement from using a near full batch policy over a greedy policy. This is consistent with other research in this area. A graph of this experiment can be viewed on our website, and includes curves for the other thresholds between 1 and 7. (You can see the graph at http://www.fabtime.-com/ctbatch.htm#mbssingle.)

C. Full Fab Model Simulation Results

Now, you might wonder if this has any effect on the factory as a whole. After all, an extra few hours here or there on the furnaces could be lost in the noise relative to the overall cycle time. We therefore did another experiment using a simplified version of a full factory model. The model had two products, 115 steps per process flow, 22 tool groups, and 21 operator groups. We simulated this model for two years, varying the start rate to allow different levels of bottleneck utilization for each run. Here are the results:

Util: 30%. Greedy: 2.1. Near-Full: 3.5
Util: 40%. Greedy: 2.2. Near-Full: 3.2
Util: 50%. Greedy: 2.3. Near-Full: 3.0
Util: 60%. Greedy: 2.4. Near-Full: 2.9
Util: 70%. Greedy: 2.7. Near-Full: 3.0
Util: 80%. Greedy: 3.3. Near-Full: 3.5
Util: 90%. Greedy: 5.2. Near-Full: 5.2

In the full factory model, the average cycle time is almost 70% greater for the full batch policy than for the greedy policy at very low utilizations. Up to 80% loading, the greedy batch policy yields lower cycle times. For very highly loaded fabs the full batch policy yields essentially the same results as the greedy policy. These results are available in graphical format on our website, at http://www.fabtime.com-/ctbatch.htm#mbslowmix.

For a more extreme example of the impact of batching on this fab, we modified the factory to have a greater number of products. We held the total volume the same, but divided it among seven products instead of two. All products used the same process flow, but for certain batch tools in the model, lots of different product types could not be batched together. This change thus increased the volume of distinct batch IDs in the model. The change led to a slight degradation in performance under the greedy policy, and to a significant cycle

time increase under a full batch policy, as follows:

Util: 30%. Greedy: 2.6. Near-Full: 6.8
Util: 40%. Greedy: 2.8. Near-Full: 5.5
Util: 50%. Greedy: 2.9. Near-Full: 4.9
Util: 60%. Greedy: 3.1. Near-Full: 4.5
Util: 70%. Greedy: 3.5. Near-Full: 4.6
Util: 80%. Greedy: 4.1. Near-Full: 5.0
Util: 90%. Greedy: 6.3. Near-Full: 6.4

Clearly, batching policy makes a big difference in this high-product mix fab (see graph at www.fabtime.com/-ctbatch.htm#mbshighmix), because there are so many distinct batch IDs. Lots almost always wait a long time to form a batch under a full batch policy, especially for very low utilizations. The increase in cycle time between this case and the previous case also illustrates how sensitive fab models can be to batching rules (in this case, decisions about which types of lots can be batched together).

A Potential Rule of Thumb for Look-Ahead

The above examples show that in general a greedy policy is better in terms of cycle time than a full batch policy. However, despite this general rule, there are sometimes specific cases where it makes sense, in terms of cycle time, to wait for the next lot before starting the batch. For example, suppose that one lot is available for processing, and another lot of the same recipe is due to arrive in five minutes, while the process time for the batch is 24 hours. Clearly, it makes sense to wait for that second lot before starting the batch. However, the situation is usually less straightforward. Suppose that you have a furnace that can hold six lots, and four lots of the same recipe are ready to be processed, with a process time of eight hours. Now suppose that another lot of the same recipe is due to arrive in an hour. Should you wait for the fifth lot? Perhaps not. What if the process time is 24 hours? Does that make a difference? In this section, extracted from the main article in Issue 3.08, we outline a simple rule for deciding when to wait for the next lot, and when to just start the batch.

The Simplest Possible Case

Starting with the simplest possible case, we have:

- Single furnace
- Batch time = *ProcessTime*
- Lots ready to go = *NumberReady*
- Waiting for *NumberIncoming* lots that are due to arrive in *WaitingTime* hours (batch arrivals)

If we wait for the lots that are coming, then it will save those lots (*ProcessTime-WaitingTime*)**NumberIncoming* hours of cycle time.

However, waiting for the incoming lots will add *WaitingTime* * *NumberReady* hours of cycle time for those that are ready to go.

If we are comparing on the basis of average cycle time = (Lot1CycleTime + Lot2CycleTime + ...) / (*NumberReady* + *NumberIncoming*), then it's equivalent to compare on the basis of total cycle time = (Lot1CycleTime + Lot2CycleTime), because (*NumberReady* + *NumberIncoming*) is the same whether we start the batch now or whether we start the batch after *WaitingTime*.

Since we are adding the numbers together, we just need to know if the amount saved by waiting is more than the amount delayed by waiting, e.g.

If

TimeSaved = (ProcessTime-WaitingTime)*NumberIncoming

is greater than

TimeDelayed = WaitingTime * NumberReady

then it makes sense to wait for the incoming lot(s).

For example, suppose we have:

ProcessTime = 8 hours to process a batch (regardless of batch size)

NumberReady = 4 lots ready to process right now

NumberIncoming = 1 lot arriving soon

WaitingTime = 2 hours until the incoming lot is due to arrive

If we wait for the one incoming lot, then *TimeSaved* = (*ProcessTime-WaitingTime*)**NumberIncoming* = (8 hours - 2 hours)*1 lot = 6 hours, and *TimeDelayed* = *WaitingTime* * *NumberReady* = (2 hours) * 4 lots = 8 hours. In this case, *TimeSaved* = 6 hours < *TimeDelayed* = 8 hours, and so it does not make sense to wait for the additional lot. If, however, *NumberIncoming* was 2, then *TimeSaved* would be 6 hours times 2, or 12 hours, and then it would make sense to wait.

Extensions and Other Considerations

The above formula does not take into account the fact that NumberIncoming + NumberReady might be greater than the capacity of the furnace. Nor does it account for situations where there is a second furnace that can be used by the incoming lots, or the case where one lot is arriving in WaitingTime minutes, and another shortly thereafter. It could also be extended to take into account the state of a downstream serial machine (e.g. it might be better to process now than to overload the downstream tool). These considerations are discussed in a bit more detail in Issue 3.08. The formulas get fairly involved when these other aspects are taken into account, but could certainly be coded into your dispatching system. The main point is that it's not enough to consider the time saved for the incoming lot by waiting for it. You have to compare that to the cumulative waiting time incurred for all lots that are already there.

It should also be noted that any benefit from waiting is jeopardized if there is inaccuracy in the forecasts of when lots are due to arrive. The worst thing is to wait for a particular lot, and then have it get held up, and not arrive for two extra hours. Therefore, this type of rule makes the most sense for fabs that have relatively stable data regarding when future lots are going to arrive. Research has shown that in the presence of inaccuracy in arrival forecasts, the benefits to using look-ahead are very slight.

Conclusions

Batch processing is a significant source of variability, and hence of cycle time, in most wafer fabs. Smaller, more frequent batch runs are better for cycle time than large, infrequent runs. A full batch policy, by which a fab waits until a full load is available before running a batch tool, can significantly drive up cycle times for tools that aren't heavily loaded. A greedy policy, by contrast, yields better cycle times at low to moderate utilizations, and usually has no cycle time penalty at high utilizations. Of course there are times when it makes sense to run full, or nearly-full, batches, if dictated by cost, equipment maintenance, or other considerations. However, for cycle time it is better to apply a greedy policy than a full batch policy for each batch tool (or at least a modified greedy policy where you always wait for two or three lots, but don't wait for the full batch). On top of that base greedy policy, there may still be situations in which it makes sense to wait for one or more soon-to-arrive lots. In the article above we have outlined a simple rule of thumb for deciding when to wait vs. when to go ahead and start the batch. While this rule would need to be modified in practice, to account for more of the complexities of wafer fabs, the general philosophy remains the same: compare the time that you incur by waiting for incoming lots to the time that you delay lots that are already there, and decide accordingly.

Closing Questions for FabTime Subscribers

Have you implemented a greedy batching policy in your fab? Have you tried look-ahead? If so, do you find that forecast

accuracy is good enough for your look-ahead policy to help? Do you use simulation to test out the impact of batching policies? We welcome your feedback.

Further Reading

For a much more detailed discussion on batch size decision rules, with references to the theory underpinning these results, see "A Review of Real-Time Control Strategies for Furnace Batch Sizing in Semiconductor Manufacturing" by J. K. Robinson, J. W. Fowler, and J. F. Bard, available from www.fabtime.com/abs_MfgRev.htm.

Other articles that discuss batch size decisions in wafer fabs include:

- E. Akçali and R. Uzsoy (Purdue University) and D. G. Hiscock, A. L. Moser, and T. J. Teyner (Intersil), "Alternative Loading and Dispatching Policies for Furnace Operations in Semiconductor Manufacturing: A Comparison by Simulation," *Proceedings of the 2000 Winter Simulation Conference*, 2000.
- J. W. Butterbaugh, "Strategies for Cycle Time Reduction in Batch Cleaning," *IEEE 2004 Advanced Semiconductor Manufacturing Conference (ASMC '04)*, 52-56, 2004.
- Noah Chiou, Ivan Wang, Jerry Chang, and Topas Chang, "The Tool Efficiency Monitoring System Creation of Furnace Area in Semiconductor Manufacturing," *Proceedings of the 2002 Semiconductor Manufacturing Technology Conference*, 132-135, 2002.
- J. W. Fowler, G. L. Hogg, and D. T. Phillips, "Control Of Multiproduct Bulk Server Diffusion/Oxidation Processes. Part 2: Multiple Servers," *IIE Transactions*, Vol. 32, No. 2, 167-176, 2000.
- J. W. Fowler, N. Phojanamongkolkij, J. K. Cochran, D. C. Montgomery, "Optimal Batching in a Wafer Fabrication Facility Using a Multiproduct G/G/c Model with Batch Processing," *International Journal of Production Research*, Vol. 40, No. 2, 275-292, 2002.
- C. R. Glassey and W. W. Weng, "Dynamic Batching Heuristic for Simultaneous Processing," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 14, No. 2, pp. 77-82, 1991.
- H. Gurnani, R. Anupindi, and R. Akella, "Control of Batch Processing Systems in Semiconductor Wafer Fabrication Facilities," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 5, No. 4, pp. 319-328, 1992.
- K. Ibrahim, M. A. Chik, W. S. Nizam, N. L. Fern, and N. F. Za'bah, "Efficient Lot Batching System for Furnace Operation," *IEEE 2003 Advanced Semiconductor Manufacturing Conference (ASMC '03)*, 175-187, 2003.
- Lars Mönch and Ilka Habenicht (Technical University of Ilmenau), "Simulation-Based Assessment of Batching Heuristics in Semiconductor Manufacturing," *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., 2003.
- J. K. Robinson, J. W. Fowler, and J. F. Bard, "The Use of Upstream and Downstream Information in Scheduling Semiconductor Batch Operations," *International Journal of Production Research*, Vol. 33, No. 7, 1849-1870, 1995. (abstract at www.fabtime.com/abs_IJPR.htm).
- L. Solomon, J. W. Fowler, M. Pfund, and P. H. Jensen, "The Inclusion of Future Arrivals and Downstream Setups into Water Fabrication Batch Processing Decisions," *Journal of Electronics Manufacturing*, Vol. 11, No. 2, 149-159, 2002.
- W. W. Weng and R. C. Leachman, "An Improved Methodology for Real-Time Production Decisions at Batch-Process Work Stations," *IEEE Transactions on Semiconductor Manufacturing*, 1993.

Subscriber List

Total number of subscribers: 2812, from 475 companies and universities. 22 consultants.

Top 20 subscribing companies:

- Maxim Integrated Products, Inc. (241)
- Intel Corporation (155)
- Micron Technology, Inc. (82)
- Analog Devices (66)
- Infineon Technologies (64)
- X-FAB Inc. (63)
- Freescale Semiconductor (61)
- NEC Electronics (61)
- Texas Instruments (60)
- STMicroelectronics (59)
- Cypress Semiconductor (55)
- International Rectifier (55)
- ATMEL (54)
- Chartered Semiconductor Mfg. (54)
- TECH Semiconductor Singapore (54)
- ON Semiconductor (53)
- NXP Semiconductor (48)
- IBM (46)
- Spansion (36)
- Seagate Technology (32)

Top 3 subscribing universities:

- Virginia Tech (11)
- Ben Gurion Univ. of the Negev (7)
- Nanyang Technological University (7)

New companies and universities this month:

- AAI Corporation
- The Boeing Company
- DEE - Politecnico di Bari
- Garmin

Note: Inclusion in the subscriber profile for this newsletter indicates an interest, on the part of individual subscribers, in cycle time management. It does not imply any endorsement of FabTime or its products by any individual or his or her company.

There is no charge to subscribe and receive the current issue of the newsletter each month. Past issues of the newsletter are currently only available to customers of FabTime's web-based digital dashboard software or cycle time management course.

To subscribe to the newsletter, send email to newsletter@FabTime.com, or use the form at www.FabTime.com/newsletter.htm. To unsubscribe, send email to newsletter@FabTime.com with "Unsubscribe" in the subject. FabTime will not, under any circumstances, give your email address or other contact information to anyone outside of FabTime without your permission.

FabTime® Cycle Time Management Software

"Instead of spending time preparing reports, shift facilitators can get the data they need quickly from FabTime, and then spend their time making real improvements."
Mike Hillis
Cycle Time and Line Yield Improvement Manager
AMD Fab 25

FabTime Subscription

One low monthly price includes

- Software installation and real-time connect to your MES
- End user and system administrator training
- Unlimited users via your Intranet.
- Software maintenance and regular upgrades (approx. 6 per year, via our no-downtime patch system)
- Add-on dispatching and planning module for a slightly higher monthly fee

Interested?

Contact FabTime for technical details or a pilot project quote.

FabTime Inc.
Phone: +1 (408) 549-9932
Fax: +1 (408) 549-9941
Email: Sales@FabTime.com
Web: www.FabTime.com

Turn fab MES data into information and save time and money

- Are your supervisors swamped with daily reports, but lacking real-time information?
- Is it difficult to link equipment performance to cycle time?
- Does each new cycle time analysis require IT resources?

FabTime can help. FabTime saves your management team time daily by turning fab MES data into information, via a real-time web-based dashboard that includes lot dispatching. FabTime saves your IT staff time by breaking the cycle of custom-developed reports. With FabTime, the end user can filter for exactly what he or she needs, while staying in a comprehensive framework of pre-defined charts. Most importantly, FabTime can help your company to increase revenue by reducing cycle times up to 20%.

"I use FabTime every day, and so do the supervisors who report to me. The data that I need is right on my home page where I need it when I come in every morning."
Jim Wright
Production Manager
Headway Technologies

FabTime Benefits

- Cut cycle times by up to by 20%.
- Focus improvement efforts on the tools that inflate cycle time.
- Improve supervisor productivity – cut reporting time by 50%.