Control of Job Arrivals with Processing Time Windows into Batch Processor Buffer

John Benedict C. TAJAN[1], Appa Iyer SIVAKUMAR[1], and Stanley B. GERSHWIN[2]
[1] Nanyang Technological University
[2] Massachusetts Institute of Technology

Abstract **Consider a two-stage manufacturing system composed of a batch processor and its upstream feeder processor. Jobs exit the feeder processor and join a queue in front of the batch processor, where they wait to be processed. The batch processor has a finite capacity *Q*, and the processing time is independent of the number of jobs loaded into the batch processor. In certain manufacturing systems (including semiconductor wafer fabrication), a processing time window exists from the time the job exits the feeder processor till the time it enters the batch processor. If the batch processor has not started processing a job within the job's processing time window, the job cannot proceed without undergoing rework or validation by process engineers. We generalize this scenario by assigning a reward *R* for each successfully processed job by the feeder processor, and a cost *C* for each job that exceeds its processing time window without being processed by the batch processor. We examine a problem where the feeder processor has a deterministic processing time and the batch processor has stochastic processing time, and determine that the optimal control policy at the feeder processor is insensitive to whether the batch processor is under no-idling or full-batch policy.**

Index Terms **Wafer Fabrication, Optimal Control, Batch Processor, Processing Window**

I. INTRODUCTION

An ignored facet of controlling diffusion furnaces in wafer fabs is the presence of processing time windows for jobs that are queuing in front of the batch processor. Upon exiting the processor immediately upstream of the batch processor (which we call the feeder processor), a job is deemed to be contaminated if it has waited in front of the batch processor for at least *T* time units; this job may need to be reprocessed, or revalidated by process experts before the job can continue to be processed. We refer to this scenario as the job exceeding its processing time window at the batch processor. This is unwanted for several reasons. Firstly, a furnace's feeder processor frequently also serves as the feeder processor for a different processor *M*. Each job that exceeds its processing time window is equivalent to lost throughput for processor *M*, without increasing the furnace's throughput. Secondly, a job that exceeds its processing time window incurs the costs of rework, or validation by process engineers. Increased rework or validation also increases the cycle time for at least the expired jobs, which is undesirable, due to the larger yield losses correlated with higher cycle times. Figure 1 shows a subsection of a wafer fab process flow involving the furnace and its feeder processors. The furnace can have more than one feeder processor, and the feeder processors typically feed more than one processor.

Manuscript received November 20, 2006. This work was supported by the Singapore-MIT Alliance (SMA), Singapore.

J. C. Tajan is a PhD student with the Singapore-MIT Alliance under the Innovations in Manufacturing Systems and Technology program (phone: (65)-9190-8354; e-mail: taja0001@ ntu.edu.sg)

A. I. Sivakumar is an Associate Professor with the Mechanical and Aerospace Engineering Department, Nanyang Technological University, Singapore. (e-mail: msiva@ntu.edu.sg)

S. B. Gershwin is a Senior Research Scientist with the Mechanical Engineering Department, Massachusetts Institute of Technology, MA, USA. (e-mail: sbgershwin@mit.edu).

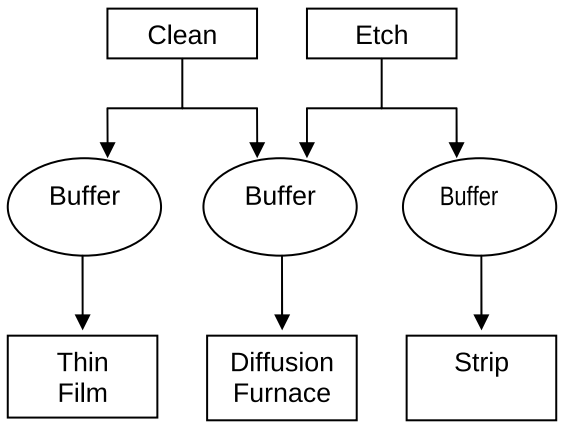

Figure 1: A subsection of a process flow inside the wafer fab. The diffusion furnace is fed work-in-progress by either Clean or Etch. However, both Clean and Etch can also process jobs to feed other processors. The Clean station can also process jobs for Thin Film, while Etch can also process jobs for Strip.

The control of the feeder processor, in accordance with the existence of processing time windows at the batch processor is a problem that has been largely ignored by researchers. We address the problem of controlling job flow into the batch processor queue, considering the processing time window present in jobs.

There are other systems in which processing time windows are present. Processing time windows also exist in manufacturing and transport systems where the product is perishable, for example. When there is a considerable amount of products waiting to be transported, it may be optimal for the manufacturing facility to stop processing products with finite shelf life.

II. Literature Review

Batch processors, as covered in existing literature, can be broadly classified into two types: batch processors with compatible jobs, and batch processors with incompatible jobs. For batch processors with compatible jobs, the composition of a batch is constrained only by the capacity of the batch processor; however, the processing time of the batch processor is set to be the maximum of the minimum processing time requirements for all the jobs inside a particular job. The burn-in oven used to simulate repeated usage of semiconductor chips is a good example of batch processors with compatible jobs. On the other hand, diffusion and oxidation ovens are batch processors with incompatible jobs. Each job belongs to a particular job family, and only jobs from the same family can be batched together. Processing time of the batch is only dependent on the job family being processed. For this problem, we make the simplifying assumption that all jobs belong to the same job family. For brevity, we will simply refer the interested reader to a review of batch processor control literature by [Mathirajan and Sivakumar 2006].

If we know the arrival times of each job into the batch processor queue, then it is possible to incorporate processing time windows into the analysis by assuming a processing sequence at the feeder processor, and if feeder processor processing times are deterministic, the arrival times for each job to the queue in front of the batch processor can be generated. The due times for each job can then be generated, and we can then minimize the number of tardy jobs.

To the best of our knowledge, only [Makis 1985] has explicitly considered the processing time window of jobs that are being processed by a batch processor. He looked into the problem of controlling a batch processor with infinite capacity. The system has an affine service cost function, and the holding cost function is nonnegative. Jobs have to be processed within a processing time window. He determined that the optimal policy is a threshold policy, and proceeds to examine the steady-state characteristics of the system under the optimal policy.

Our work differs from what has been done primarily in the location of the control. Previous research treats the arrival times of jobs to be a constraint, and attempt to optimize the batch processor control policy given this set of constraints. We looked into controlling the feeder processor directly upstream of the batch processor. This is because even an optimal policy at the batch processor can perform badly if there is an extremely large amount of jobs queued up in front of it, due to a sub-optimal feeder control policy.

III. Problem Statement

A two-stage manufacturing system is comprised of an upstream feeder processor feeding a downstream batch processor via a buffer of infinite capacity. The buffer level does not include the jobs that are being processed by the batch processor. In wafer fabrication, the downstream batch processor corresponds to a diffusion oven, while the feeder processor can be either a serial processor (etch station) or a batch processor with capacity two (clean station). We initially assume that the feeder processor is a serial processor. The batch processor can process up to Q jobs at a time, and its processing time is independent of the number of jobs loaded into the batch processor. Upstream of the serial processor is an infinite source of jobs, and downstream of the batch processor is an infinite sink for jobs. Both processors cannot be preempted. See Figure for an illustration of the manufacturing system model. The manufacturing system is assumed to operate on a discrete time scale.

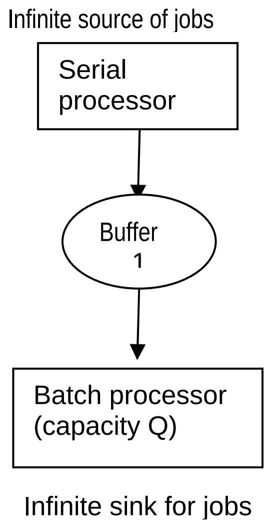

Figure 2: Simple 2-stage manufacturing model. Without a cost associated with jobs exceeding their processing time windows, the optimal policy at the feeder processor (in this case, the feeder processor is initially assumed to be a serial processor) would be to process a job at each instance the feeder processor is available.

There is a processing time window between the batch processor and its feeder processor. If a job has not started processing at the batch processor T time units after it has exited the feeder processor, a cost C is incurred. This cost C quantifies lost throughput at the feeder processor, yield loss due to longer cycle time and other ancillary effects of rejecting a job. For each job that the feeder processor processes (including jobs that end up exceeding the processing time window), a reward R is generated. This reward is the incremental benefit of processing a job for the batch processor, as compared to feeding a different processor. We wish to determine when it is appropriate to process at the feeder processor, given the number of jobs in front of the batch processor, and the status of the batch processor. It is assumed that $C>R$, otherwise, it would be always profitable to process a job, regardless of the probability of the job exceeding its processing time window.

IV. Problem Methodology

We look at the probability that the prospective job at the serial processor will have to queue for at least T time units in front of the batch processor before it can start processing at the batch processor. We obtain expressions for the expected reward of processing a job, and stop processing at the feeder processor when it is no longer profitable to process a job, given the reward R and the possibility of incurring a cost C.

A. Assumed Batch Processor Control Policies

We assume that the batch processor is operated according to two policies:

a.) no-idling policy – whenever the batch processor is idle and there are jobs waiting in front of the batch processor, the batch processor will immediately start a batch, regardless of the number of jobs that can be processed. This is equivalent to a minimum batch size (MBS) policy with $X=1$.

b.) full-batch policy –a batch is started only if the batch processor is idle and a full batch can be processed. This is equivalent to an MBS policy with $X=Q$.

Analysis of the problem when the batch processor is under full-batch policy entails estimating when additional jobs after the prospective job is processed will arrive. We make the assumption that the feeder processor will continue to process jobs until it is told to stop.

V. Model One Assumptions and Development

The following assumptions are made in model one:

- Both machines perfectly reliable
- Serial processor processing time is deterministic. It takes one time unit to process one job at the serial processor.
- Batch processor processing time is geometrically distributed. Probability (batch being finished in current time instance) = P.

At any time instance t, let Z_t be the current buffer level in front of the batch processor, and S_t be the current state of the batch processor, with one meaning the batch processor is busy and zero meaning the batch processor is idle. Then, the state of the system X_t can be described as $X_t = (Z_t, S_t)$. The serial processor takes one time unit to process a job. We can omit tracking the state of the serial processor, as it is guaranteed to be available at the start of each time instance.

A. When the batch processor is under no-idling policy

We divide the model development into three different cases, based on the state of the system.

Case 1: $X_t = (0, 0)$

The prospective job at the serial processor will have zero queue time. Once it enters the batch processor queue, it is immediately processed by the batch processor. In this case, it is profitable to process the job.

Case 2: $X_t = (0 \text{ to } Q-1, 1)$

We process the prospective job only if $R>C$*Probability (current batch will take at least $T+1$ units to process).

We process the prospective job only if: $R/C > (1-p)^T$

(1)

Case 3: $X_t = (\geq Q, 1)$

Let $\lfloor x \rfloor$ be the largest integer less than or equal to x. (Conversely, let $\lceil x \rceil$ be the smallest integer greater than or equal to x.) The number of batches the batch processor needs to process before the prospective job can be loaded is $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$. The prospective job should be processed only if: $R > C * \Pr obability(\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$ *batches will take at least T+1 units to process).* The probability that $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$ batches will be processed in exactly x time units has a Pascal distribution. We sum up the probabilities of $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$ batches being processed in exactly x time units, for $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1 \leq x \leq T$ to obtain the probability of not incurring a cost, and this is subtracted from unity to obtain the probability of incurring a cost.

Let $^{x}C_{y} = x! / (y!)(x-y)!$, then

$$R/C > 1 - \sum_{i=\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}^{T} ({}^{i-1}C_{\left\lfloor \frac{Z_t}{Q} \right\rfloor})(P)^{\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}(1-P)^{i - \left\lfloor \frac{Z_t}{Q} \right\rfloor - 1}.$$

(2)

$T > \left\lfloor \frac{Z_t}{Q} \right\rfloor$ is a necessary condition for the prospective job to have a positive probability of avoiding a cost. Since the system runs in discrete time, it will take at least $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$ time units to process all the batches in front of the incoming job (including the batch currently being processed by the batch processor). The serial processor takes one time unit to process the batch. The prospective job will be joining batch $\left\lfloor \frac{Z_t}{Q} \right\rfloor + 2$.

B. *When the Batch Processor is under Full-Batch Policy*

When the batch processor is under full-batch policy, it is only profitable to process a full batch if the time it takes to form a full batch will not cause a job to exceed its processing time window *T*. Thus, a necessary condition for the full-batch policy to be profitable is $T \geq Q-1$. If $T<Q-1$, then it is not profitable to form a full batch, regardless of how many jobs are queued up in front of the batch processor. $T \geq Q-1$ will be assumed throughout the three cases.

We consider three different cases:

Case 1: $X_t = (0 \text{ to } Q-1, 0)$

The prospective job will have queue time equal to the time it takes to form a full batch. Given our additional assumption that $T \geq Q-1$, then we will always choose to process under case 1.

Case 2: $X_t = (0 \text{ to } Q-1, 1)$

The prospective job can incur processing time in two ways. Firstly, the current batch may not be finished before the job arrives at the buffer. Secondly, the job may be forced to wait for additional jobs to arrive in order to form a full batch.

Prospective job's waiting time = MAX (processing time of the current batch-prospective job's processing time at the serial processor, time prospective job has to wait to form a full batch)

Prospective job should be processed only if:

$T \geq Q-Z_t-1$ (this is covered by previous assumption on the profitability of the full-batch policy) AND *R>C*Probability (current batch will take at least T+1 time units to process).*

Prospective job should be processed only if:

$$R/C > (1-p)^T \qquad (3)$$

Case 3: $X_t = (\geq Q, 1)$

Waiting time for prospective job = MAX (processing time of all full batches ahead of prospective job-prospective job's processing time at the serial processor, time prospective job has to wait to form a full batch). Let B_t be the number of additional jobs the prospective job will have to wait for to form a full batch. Let $(Z_t \setminus Q) = Z_t - \left\lfloor \frac{Z_t}{Q} \right\rfloor * Q$. If $(Z_t \setminus Q)=0$, then the prospective job starts a new batch, and $B_t=Q-1$. Otherwise, $B_t=Q-(Z_t \setminus Q)-1$. Prospective job should be processed only if:

$T \geq B_t$ AND $R > C * probability(\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1$ batches will take at least $T+1$ units to process)

$$R/C > 1 - \sum_{i=\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}^{T} \left({}^{i-1}C_{\left\lfloor \frac{Z_t}{Q} \right\rfloor}\right)(P)^{\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}(1-P)^{i-\left\lfloor \frac{Z_t}{Q} \right\rfloor - 1} \qquad (4)$$

VI. Model One Analysis

For model one, the optimal policy will be identical regardless of whether the batch processor policy is no-idling or full-batch, assuming that $T \geq Q-1$. The optimal policy is:

If the batch processor is idle, always process the prospective job

Else

If there is less than a full batch in front of the batch processor and the batch processor is busy, process the prospective job if $R/C > (1-p)^T$. If this condition is not met, then no job should wait in front of the batch processor when the batch processor is not idle.

Else

there is at least one full batch in front of the batch processor. Process the prospective job if:

$$R/C > 1 - \sum_{i=\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}^{T} \left({}^{i-1}C_{\left\lfloor \frac{Z_t}{Q} \right\rfloor}\right)(P)^{\left\lfloor \frac{Z_t}{Q} \right\rfloor + 1}(1-P)^{i-\left\lfloor \frac{Z_t}{Q} \right\rfloor - 1}.$$

The optimal policy at the feeder processor is a threshold policy. This is because the probability of a prospective job incurring a queuing time of at least *T* time units is a non-decreasing function with respect to the number of jobs in front of the batch processor. Figure 3 illustrates a typical graph of the probability of the prospective job incurring a cost, with respect to the number of jobs in front of the batch processor. The probability of a job incurring a cost is a step function, with respect to the number of batches in front of it. If we know that it is profitable to process a job when there are *n*Q* jobs in the buffer (this means that the prospective job will end up starting a new batch), then we immediately know that it is also profitable to process a job when the number of jobs in the buffer are from *n*Q+1 to (n+1)*Q-1*. This simplifies the search for the threshold value as we only need to test for threshold values that are integer multiples of Q.

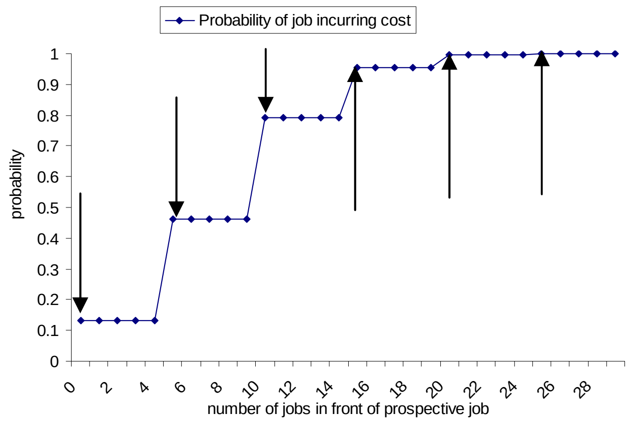

Figure 3: Probability of the prospective job incurring a cost when the batch processor is under no-idling policy, $T=5$, $Q=5$ and $P = 1/3$, assuming that the batch processor is already busy. When the batch processor is idle, the probability of incurring a cost is zero. The arrows point to the instances when the prospective job will form a new batch. When the job forms a 6th batch queuing in front of the batch processor, the probability of incurring a cost becomes 1. Since the batch processor is busy, the system will have to process 6 batches in $T=5$ time instances for the prospective job not to incur a cost, and this is not possible, due to the problem assumptions.

VII. Extensions to Other Models

Two qualities of the optimal policy for Model 1 are highly desirable: (1) the optimal threshold is an integer number of the batch processor capacity, and (2) the optimal policy is insensitive to the batch processor policy. Unfortunately, both of these properties no longer hold when the assumptions are slightly changed. For example, when the processing time of the serial processor is deterministic but larger than one, both properties are no longer true. This is further discussed in other work.

VIII. Conclusion

An important subsystem inside the wafer fab is the oxidation/diffusion oven, a batch processor, and its feeder processor. An important facet of scheduling systems with batch processors that exists in wafer fabs is the existence of processing time windows. Jobs exiting the feeder processor typically have to be processed by the batch processor within a certain period of time after it has exited the feeder processor. Jobs that exceed their processing time windows may have to undergo rework or validation, due to contamination concerns. We generalize this scenario by associating a reward R with each job processed by the serial processor, and a cost C with each job that exceeds its processing time window. We use a two-stage model, with a serial processor feeding a batch processor via an infinite buffer, and assume this two-stage system has an infinite source and an infinite sink.

Problem 1 assumed a deterministic serial processor processing time, with a geometrically distributed batch processor processing time. We derived methods to obtain the optimal control policy at the feeder processor, when the batch processor is under either a full-batch or a no-idling policy. Furthermore, we have shown that the optimal control policy is a threshold policy, with the threshold being an integer multiple of the batch processor capacity. Given a profitability assumption on the full batch policy, the optimal policy is identical for the two batch processor control policies. Unfortunately, these properties rarely remain true when the model assumptions are changed. For example, when the serial processor processing time remains deterministic but becomes larger than one, both properties no longer hold true. This is looked into in further detail in future work.

References

[1] Mathirajan M. and Sivakumar A.I., *"A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor,"* International Journal of Advanced Manufacturing Technology **29** (2006) 990-1001

[2] Makis, V., *"Optimal control of a batch service queueing system with bounded waiting time,"* Kybernetika 21 (1985) 262-271