EFFICIENT SCHEDULING POLICIES TO REDUCE MEAN AND VARIANCE OF CYCLE–TIME IN SEMICONDUCTOR MANUFACTURING PLANTS*†

Steve C. H. Lu, Deepa Ramaswamy and P. R. Kumar‡

Abstract

We address the problem of reducing the mean and variance of cycle–time in semiconductor manufacturing plants. Such plants feature a characteristic re–entrant process flow, where lots repeatedly return at different stages of their production to the same service stations for further processing, consequently creating much competition for machines.

We introduce a new class of scheduling policies, called Fluctuation Smoothing policies. Unanimously, our policies achieved the best mean cycle–time and standard deviation of cycle–time, in all the configurations of plant models and release policies tested. As an example, under the recommended Workload Regulation Release policy, for a heavily loaded Research and Development Fabrication Line model, our Fluctuation Smoothing policies achieved a reduction of 22.4% in the mean queueing time, and a reduction of 52.0% in the standard deviation of cycle–time, over the baseline FIFO policy.

These conclusions are based on extensive simulations conducted on two models of semiconductor manufacturing plants. The first is a model of a research and development fabrication line that has been developed and studied earlier by Wein [1]. The second is an aggregate model intended to approximate a full scale production line. Statistical tests are used to corroborate our conclusions.

*Please address all correspondence to the third author at the address below. Tel. (217)-333-7476.

†The research reported here has been supported in part by the National Science Foundation under Grant Nos. NSF–ECS–90–25007 and NSF–ECS–92–16487, and in part by the Joint Service Electronics Program under Contract No. N00014–90–J–1270.

‡Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois, 1308 West Main Street, Urbana, IL 61801.

1 Introduction

Semiconductor manufacturing requires repetitive use of several similar processing operations. The machines performing these operations are expensive. The economic necessity of reducing the capital outlay dictates that such expensive machines be shared by all lots requiring the particular processing operation provided by the machine, even though they may be at different stages of their manufacturing life. This results in a "re–entrant" flow, see Figure 1, characteristic of both semiconductor manufacturing as well as thin film lines, that differentiates them in several important ways from both traditional flow shops as well as job shops.

Typically, the sum of the processing times of the various manufacturing operations required is of the order of days. However, the manufacturing lead–time, which also includes the time spent by the lots in just waiting for service at machines, is of the order of weeks. This manufacturing lead–time, the time elapsed from the release of a lot into the plant till its emergence as a finished product, is called the "cycle–time" in semiconductor manufacturing parlance. The ratio $\frac{\text{Mean cycle–time}}{\text{Sum of processing times}}$, called the "actual–to–theoretical ratio," may range between 2.5 and 10.

It is obviously of great economic importance to reduce the mean cycle–time. For device prototyping, typically involving several design changes, a shorter product development time allows a quicker response to rapidly changing market needs. For production lines, a smaller cycle–time improves the ability to satisfy customer requirements. Also, for the same level of throughput, a shorter cycle–time results in a smaller work–in–process that not only reduces the capital tied up, but also leads to an uncluttered plant floor. Finally, the smaller the cycle–time, the smaller is the inventory buffer that needs to be maintained at the downstream end of the plant. When product designs become obsolescent, such inventory may lose value. There is also a technological reason for reducing the cycle–time. The shorter the period that wafers are exposed to aerial contaminants while waiting for processing, the smaller is the yield loss.

It is also important to reduce the *variance* of the cycle–time. It allows a more accurate

prediction of production completion time, which facilitates improved downstream coordination of further operations on completed wafers, such as assembly, etc.

There are primarily two ways in which control is exercised over the plant. First, one can specify when new lots are to be released into the plant. This is done by the *release policy*. Clearly, the release policy must meet some constraints such as maintaining an average release rate of lots. Second, for lots already in the plant, one has to decide which lot is processed next at each machine as it becomes available. This is done by the machine or lot scheduling policy, which we call the *scheduling policy*.

Many previous approaches to release and scheduling attempt to deal with the bottlenecks. We refer the reader to Wein [1], Harrison and Wein [2], Glassey and Resende [3], and Lozinski and Glassey [4], for an account of these approaches. Some approaches are rooted in heavy traffic Brownian network theory, see [1, 2] and more recently Chevalier and Wein [5], and the references cited there. More generally Uzsoy, Lee and Martin–Vega [6, 7] provides a survey of the problems of scheduling semiconductor manufacturing plants, and Kumar [8] an account of some of the theory of re–entrant lines.

The manner in which the competition for machines by lots is resolved in large re–entrant lines, has a clear bearing on plant performance measures such as mean and variance of the cycle–time. That is the thesis of this paper. We propose a new approach of smoothing the fluctuations in *all* the flows in the network. Thus, we do not restrict attention to just the bottleneck machines in the plant. We develop a new class of scheduling policies called *Fluctuation Smoothing Policies* (FS), which can substantially reduce the mean and standard deviation of cycle–time. Unanimously, in all the plant models and release policies studied, our new FS policies provided the best mean and variance of cycle–time. As an example, our FSMCT scheduling policy, described in Section 3.3, reduced the mean queueing time by 22.4%, and the standard deviation of cycle–time by 52.0%, over the baseline FIFO policy, for a heavily loaded Research and Development Fabrication Line model with Workload Regulation Releases. This benefit is in addition to the benefit provided by the proper choice of the release policy. Our bottom line recommendation is to use the Workload Release Regulation

policy together with the Fluctuation Smoothing scheduling policy FSMCT described below.

The conclusions regarding the effectiveness of our proposed scheduling policy are based on about eleven thousand simulations conducted on two models of plants. The first is a model based on a Hewlett–Packard Research and Development Fabrication Line. It has earlier been modeled and studied by Wein [1], who performed a detailed comparison of eleven scheduling policies and four release policies. Wein's net conclusion was that among the set of scheduling policies considered by him, the mean cycle–time was not very sensitive to the choice of a scheduling policy, but could be substantially reduced by the proper choice of a release policy. For implementation work following up on these recommendations, see Miller [9]. Here, by introducing a new approach, and a new class of scheduling policies, we show that *in addition* to the benefit obtained from a proper choice of a release policy, one can obtain a substantial additional reduction, simultaneously in both the mean and variance of cycle–time, by proper choice of a scheduling policy.

The second plant considered is an aggregated model of a full scale production line. The data used was not drawn from any existing plant, and should therefore only be regarded as a gross approximation. Nevertheless, since production lines tend to be more heavily loaded and feature much greater throughput than Research and Development Fabrication lines, we felt it was useful to see if our results continued to hold for other models. As we show, we obtained similar simultaneous reductions in both the mean and variance of cycle–times.

Statistical tests have also been performed to validate our conclusions.

2 Least Slack Policies

Consider a re–entrant line as shown in Figure 2. It is convenient to suppose that the lots are stored in virtual *buffers* labeled $b_1, b_2, \ldots, b_L$, in the order that they are visited. Our new policies are a subclass of the class of Least Slack policies, see Conway, Maxwell and Miller [10], and have the following structural form. To each *lot* π entering the plant, we will associate a real number attribute $\beta(\pi)$. Also, to each *buffer* b_i, $i = 1, \ldots, L$, we will associate

a real number γ_i. If a lot π is located in buffer b_i, we define its "slack" $s(\pi)$ by,

$$s(\pi) := \beta(\pi) - \gamma_i.$$

The *Least Slack* (LS) scheduling policy gives highest priority to a lot π whose slack $s(\pi)$ is smallest. Whenever a machine located at some service station becomes idle, it scans the buffers catered to by the service station, and chooses that lot π which has the smallest slack to service next. We note that the entire class of LS policies has been proved stable in a deterministic setting; see Lu and Kumar [11] (and Kumar [8]).

3 Fluctuation Smoothing Policies

We will now develop a special class of scheduling policies that attempt to reduce various fluctuations in the queueing network. Accordingly, we call them *Fluctuation Smoothing Policies* (FS). They will turn out to be a sub–class of LS policies, featuring particular choices for $\beta(\pi)$ and γ_i.

First, in Section 3.1, we address the the problem of reducing the *variance of lateness*. Subsequently, in Section 3.2, we address the problem of reducing the *variance of the cycle–time*. Finally, in Section 3.3, we turn to the problem of reducing the *mean cycle–time*.

3.1 Reducing the variance of lateness

Suppose each lot π arriving to the plant carries a due–date,

$$\delta(\pi) := \text{ due–date of lot } \pi.$$

Also, let

$$e(\pi) := \text{ the actual time that the lot } \pi \text{ exits from the plant as finished product,}$$

and define the "lateness" $\ell(\pi)$ of a lot π by,

$$\ell(\pi) := e(\pi) - \delta(\pi).$$

Consider the problem of reducing the *variance of the lateness* of lots.

Suppose that we have been observing the plant for a long time, and have formed, for each buffer b_i, a rough estimate ζ_i of the *remaining time* that a lot π currently located in b_i is going to spend in the plant, before it exits as a finished product, i.e.,

$$\zeta_i := \text{estimate of remaining cycle–time in plant for a lot located in buffer } b_i. \tag{1}$$

If t is the current time, then $\delta(\pi) - t$ is the time remaining until lot π's due–date is up. Since ζ_i is an estimate of the time remaining until exit of lot π, the quantity $\delta(\pi) - t - \zeta_i$ measures the relative urgency of the lot π. Hence, if an idle machine has to decide which of two lots, possibly located in different buffers that it is catering, it should serve next, then it is reasonable to choose that lot π for which $\delta(\pi) - t - \zeta_i$ is smallest, i.e., the lot which is most urgent. Noting that the current time t is common across comparisons, and can thus be ignored, we can simply define the slack $s(\pi)$ of a lot π located in buffer b_i as,

$$s(\pi) := \delta(\pi) - \zeta_i. \tag{2}$$

The resulting Least Slack Policy, with $s(\pi)$ defined by (2), is in essence a "fair" policy. It attempts to make every lot equally late or equally early. Note that the *standard deviation of lateness* is small precisely when all lots are either equally late or equally early. Thus, this policy will reduce the standard deviation of lateness. We shall therefore call it the *Fluctuation Smoothing for Variance of Lateness* (FSVL) Policy.

3.2 Reducing the variance of cycle–time

Let us now turn from reducing the variance of the lateness to reducing the *variance of the cycle–time*. Let

$$\alpha(\pi) := \text{ arrival time of a lot } \pi \text{ to the plant, i.e., its release–time.}$$

Suppose now that we simply *set* the due–date of π as

$$\delta(\pi) := \alpha(\pi).$$

Then $e(\pi) - \delta(\pi) = e(\pi) - \alpha(\pi)$, and so, the lateness is the same as the cycle–time.

Thus, from the argument of Section 3.1, the choice of the slack as

$$s(\pi) := \alpha(\pi) - \zeta_i, \tag{3}$$

should lead to a reduction in the variance of the lateness. We call the resulting Least Slack Policy, as the *Fluctuation Smoothing Policy for Variance of Cycle–Time* (FSVCT).

3.3 Reducing the mean cycle–time

Let us now turn to the problem of reducing the *mean* cycle–time. It is well known in queueing theory that the delay experienced by lots at a server is caused by the burstiness of its arrivals, i.e., the variations in the interarrival times to the server, and the variations in the service times; see the Pollaczek–Khintchine Formula for M/G/1 queues, and Kingman's approximation for GI/GI/1 queues [12, 13]. Since service times are outside our control, we will try to reduce the burstiness of arrivals to each buffer. In fact, we will attempt to *simultaneously* reduce the burstiness of arrivals to *all* the buffers in the plant.

Let us consider how to reduce the burstiness in the arrivals to buffer b_{k+1}. We can do this by setting periodic due dates for *reaching* b_{k+1}. Let us denote

$$\lambda := \text{ the mean release rate (i.e., throughput).}$$

(i.e, $\frac{1}{\lambda}$ is the mean interarrival time between lots). If lot π is the n-th release into the system, we set its *due date to reach* b_{k+1} as n/λ. Then, if we reduce the variance of lateness in reaching b_{k+1}, we will obtain a traffic stream into b_{k+1} that is nearly deterministic, and hence not bursty.

Since we want to reduce the variance of the lateness in reaching b_{k+1}, we can simply regard b_{k+1} as the "exit" of the system, and then apply the policy of Section 3.1. Thus, let

$$\xi_i^k := \text{ an estimate of the delay to go from } b_i \text{ to } b_{k+1}, \text{i.e., the } \textit{remaining partial} \text{ cycle–time.}$$

Then, for a lot π in buffer b_i, which is the n-th release into the system, we define the slack as,

$$s(\pi) := \frac{n}{\lambda} - \xi_i^k. \tag{4}$$

Note that the above estimate ξ_i^k satisfies,

$$\xi_i^k = \zeta_i - \zeta_{k+1},$$

where the ζ_i's are defined as in (1). Hence the slack (4) is,

$$s(\pi) = \frac{n}{\lambda} - \zeta_i + \zeta_{k+1}.$$

However, since $k+1$ is fixed, and ζ_{k+1} is common across comparisons of various lots at buffers b_i, for $i = 1, \ldots k$, it can be dropped. It follows that we can simply define the slack as

$$s(\pi) := \frac{n}{\lambda} - \zeta_i \qquad \text{if } \pi \text{ is the } n\text{-th lot released into the plant, and it is in } b_i. \tag{5}$$

Above, we could ignore all buffers b_i for $i \geq k+1$, but that is clearly unrealistic. Hence we extend the resulting definition (5) to *all* the buffers. Very importantly, this policy is now found to be independent of b_{k+1}, and so we can use it to effectively diminish the burstiness of arrivals to *all* buffers simultaneously. Therefore, it is also a good candidate for reducing the *mean* cycle–time.

We shall call the Least Slack Policy with slacks defined by (5) and the ζ_i's chosen as in (1), the *Fluctuation Smoothing Policy for Mean Cycle–Time* (FSMCT). Note that such a policy should also have a beneficial effect on the variance of cycle–time, though not as much as the policy in Section 3.2 whose only goal was to reduce the variance of the cycle–time.

To summarize, we have developed three Least Slack Policies, called FSVL, FSVCT, and FSMCT. We have argued that these FS Policies should have the following properties.

- FSVL with slacks defined by (2) reduces the variance of lateness.
- FSVCT with slacks defined by (3) reduces the variance of cycle–time.
- FSMCT with slacks defined by (5) reduces the mean cycle–time. It should also lead to a small variance of cycle–time, but not as small as FSVCT.

Later, we shall report on the extent to which the last two hypotheses are confirmed by the detailed simulation experiments. We do not examine the FSVL policy, since the focus of this paper is on the mean and variance of cycle–time.

It should be noted that under Deterministic releases, i.e., periodic arrivals, see Table 1, one has $\alpha(\pi) = n/\lambda$. Hence, the two policies FSMCT and FSVCT coincide.

3.4 How to choose the delay estimates ζ_i

To fully specify any of the above policies, one has to specify the *parameters* $\{\zeta_i : 1 \leq i \leq L\}$ which define the policy. Once the ζ_i's have been chosen, if one implements the corresponding FS Policy, it will result in some *mean delays* from buffer b_i to exit, for every i. Call these resulting mean delays $\{\hat{\zeta}_i : 1 \leq i \leq L\}$. Above, in (1), we have required that the parameters ζ_i's themselves be chosen equal to the mean delays $\hat{\zeta}_i$'s.

Thus we have a circular specification; the parameters which determine the resulting mean delays are themselves specified by the mean delays. How are we to determine a choice of the ζ_i's which results in the $\hat{\zeta}_i$'s satisfying,

$$\hat{\zeta}_i = \zeta_i \text{ for all } i?$$

We have discovered that a simple iterative procedure works quite effectively. Such an iterative procedure is also used in Vepsalainen and Morton [14]. It employs repeated simulation. For the first simulation experiment, simply choose all the estimates of the remaining delays as 0, i.e.,

$$\zeta_i^{(0)} := 0 \text{ for all } i, \tag{6}$$

(We use the superscript "(0)" to denote the initial simulation experiment). After running a long enough simulation, we obtain empirical estimates of the mean delays from buffer b_i to exit, for each buffer b_i. Let us denote these estimates $\{\hat{\zeta}_i^{(0)}\}$.

For the next simulation run, we use the *same seed set*, but we alter the parameters employed in the policy to $\{\zeta_i^{(1)}\}$. These new parameters are chosen equal to the estimates obtained from the previous simulation run, i.e.,

$$\zeta_i^{(1)} := \hat{\zeta}_i^{(0)} \text{ for all } i. \tag{7}$$

This procedure is repeated. One uses the mean remaining delay estimates produced by the $(n-1)$-th simulation as the parameters for the n-th simulation run. The same fixed seed

set is used in all iterations, to keep the behavior of the other random events fixed. We have found that just a few iterations of this procedure, less than ten, is enough to obtain good policies.

In fact, we can do even better. First, instead of choosing the remaining delay estimates from the very last tenth simulation run as the values of $\{\zeta_i\}$, we can take the values from whichever iteration produced the *best* result. Second, one can repeat this procedure with twenty different seed sets, and then take the average of the best results over the twenty seed sets. We have employed both these procedures to select the parameters $\{\zeta_i : i = 1, \ldots, L\}$.

4 Plant descriptions

In this section, we describe the two models of wafer fabrication lines that have been tested, and specify the plant parameters. The first is a model of a Research and Development (R&D) Fabrication Line. In an exemplary study, Wein [1] has examined the comparative performance of several release policies, see Table 1, and several scheduling policies. We have replicated all his simulation results, i.e., all his fab model–release policy–scheduling policy combinations, comparing them with the new policies FSMCT and FSVCT designed by us, as well as some others which are modifications of his policies; see Table 2. The second model is that of a full scale production line. It is not based on real data, and should only be regarded as a gross approximation. There are important differences between production lines and R&D fabrication lines, and this model is radically different from the R&D line. We felt it was important to corroborate our new approach on a model of a production line too.

4.1 The Research and Development Wafer Fabrication Lines

The R&D line previously studied by Wein [1] is a single re–entrant line comprising 172 total operations at 24 different single or multi–server stations; see Table 3. We consider four versions of the Fabrication Line, called Fab 1, Fab 2, Fab 3, and Fab 3′. The only differences between Fabs 1, 2 and 3 are in the different numbers of machines at some service stations. Fab 3′ is the same as Fab 3, except that it uses a nominal release rate of 0.0212 lots/hour instead of

the nominal rate of 0.0236 used for Fabs 1, 2, and 3. Table 4 contains the detailed description of each service station. The percentage utilization of each machine at a service station is calculated by the formula, % Utilization $= [\frac{\lambda(\text{No. of visits})\ (\text{MPT})}{\text{No. of machines}} + \frac{\text{MTTR}}{\text{MTBF+MTTR}}] \times 100.$ Above, MPT is the mean processing time, MTBF is the mean time between failures, and MTTR is the mean time to repair.

Note that the machines are subject to random failures, and each failed machine requires a random repair time. The times–between–failure, times–to–repair, and processing times all have Gamma distributions. The Gamma density function has the form, $f(x) = \frac{1}{(\alpha-1)!}\theta^{\alpha}x^{\alpha-1}e^{-\theta x}$ for $x > 0$, where $\frac{\alpha}{\theta}$ is the mean of the distribution, $\frac{\alpha}{\theta^2}$ is the variance of the distribution, and α is the shape parameter. The time–between–failure and time–to–repair distributions have shape parameter 0.5. The smallness of the shape parameters models the greater uncertainty (as compared to the exponential distribution) of failure and repair times. Also, the machine failures are non–preemptive. The processing times have shape parameter 2, which corresponds to lesser randomness.

Tables 3 and 4, and the knowledge of the shape parameters, provide a full description of the plant. All the information above is drawn from Wein [1], which may be consulted for any further information regarding the specific plant.

At the release rate of $\lambda = 0.0236$ lots/hour, Fab 1 has a single bottleneck, station 14, which is highly utilized (over 90% utilization). Fab 2 has two bottlenecks, while Fab 3 has four. The reason for investigating fabrication lines with different numbers of bottlenecks is that some of the policies considered by Wein concentrate on the bottlenecks, and depend on the number of such bottlenecks.

We have studied all the release policies detailed in [1]. They are the so called Deterministic, Poisson, Closed–Loop (CL), and Workload Regulation Release policies, briefly listed in Table 1 below (drawn from [1]). In addition to all the scheduling policies considered in [1], we also consider the EDD, SRPT++, FSMCT and FSVCT policies listed in Table 2. All scheduling policies are non–idling and non–preemptive.

4.2 Model of Full Scale Production Line

The aggregated model of the full scale production line consists of 12 stations having one or more identical machines. The entire process requires 60 operations. The processing times at each service station, the times between machine failures, and the times to repair are all assumed to be exponentially distributed. Figure 1 describes the process flow, and Table 5 specifies the plant parameters. Together, the above information fully specifies the plant model.

With a mean release rate of 0.52 lots/hour, the percentage utilization is greater than 90% for most of the service stations. In such a uniformly heavily loaded plant, most of the machines are bottlenecks, and so policies which were specifically designed for a small number of bottlenecks, such as CYCLIC, STNV, and LTNV, were not considered.

4.3 Additional details of other policies

Workload Regulation release policies contain either one or two parameters to tune; see Table 1. WR1(C) for Fab 1, and WR3(C) for Fabs 3 and 3′, each have only one parameter to choose, which is chosen to attain the desired throughput. WR2(A,B) for Fab 2 has 2 parameters to choose, and we use the additional degree of freedom to minimize the mean cycle–time, while maintaining the desired throughput rate. This is done by inspection on a separate seed set. The two parameters for the Workload Balancing scheduling policy $W(a, b)$ were also similarly chosen.

For the M1-M2 and W(a,b) policies, to account for the machine failures and multiple machines at a service station, the formula $\mu_{\text{normalized}} := \frac{\mu}{sd}$ was used to calculate a *normalized* mean processing time, which was then used to calculate the expected remaining normalized processing time and normalized amount of work; see Table 2. Here $1/\mu$ is the actual mean processing time, s is the number of machines, and $d := \frac{\text{MTBF}}{\text{MTBF+MTTR}}$.

With regard to the policies LWNQ, FIFO+, and SRPT+, the system states that are required for implementing them are updated at the times of the scheduling decisions rather than at the times of lot arrivals. This remedies a problem faced in [1].

Note that the EDD and SRPT rules differ slightly, due to the phenomenon of "overtaking," possible with random processing times and parallel machines.

In implementing the buffer priority based policies, LWNQ, FGCA, SRPT, SRPT+, SRPT++, and M1-M2, we rank lots within the same buffer according to the EDD rule instead of the normal FCFS rule. We have found that this reduces the variance of the cycle times. The reason is that the EDD rule tends to minimize overtaking.

5 Design of experiments

For both the R&D and Production lines, a comparative study of the scheduling policies has been conducted through a detailed simulation experiment, followed by statistical testing. In all, a total of about eleven thousand simulation runs were performed.

First, the procedure detailed in Section 3.4 was used to select the parameters $\{\zeta_i : i = 1, \ldots, L\}$ for every FS Policy, for every fabrication line and every release policy. The FS policies were then fixed, and the seed sets used in generating the policies were never used again in the comparative simulation study.

The release rate under Deterministic and Poisson releases was 0.0236 lots/hour for the R&D Fabs 1, 2, and 3, 0.0212 lots/hour for Fab 3′, and 0.52 lots/hour for the Production Line. To approximate these throughputs, the Closed–Loop Release policies were chosen with the number of lots in Fabs 1, 2, 3, and the Production Line, regulated at 20, 28, 45 and 140, respectively. These numbers were fixed for all the scheduling policies for a fixed fabrication line.

The parameters chosen for the WR Release policies and W(a,b) scheduling policies (see Section 4.3), are listed below. The average throughput rate each obtained in the 20 simulation runs is also shown, in parentheses.

Fab 1, WR1(C) Releases: WR1(760) for FSMCT (0.02339), WR1(76) for FSVCT (0.0234), WR1(788) for FIFO (0.02343), WR1(876) for SRPT (0.02354).

Fab 2, WR2(A,B) Releases: WR2(1000,250) for FSMCT (0.023365), WR2(1000,250) for FSVCT (0.02341), and WR2(950,310) for M1-M2 (0.02347).

Fab 3, WR3(C) Releases: WR3(3500) for FSMCT (0.0232), WR3(3500) for FSVCT (0.023255), and WR3(4100) for FIFO (0.023315).

W(a,b) scheduling: W(0.9,1.05) for Fab 2 under Deterministic Release, W(0.8,1.3) for Fab 2 under Poisson Release.

Fab 3′, WR3(C) Releases: WR3(2975) for all releases, with throughput rates 0.0212 (FSMCT), 0.0211 (FSVCT), 0.0213 (CYCLIC), 0.0212 (LTNV), 0.0210 (FIFO+), 0.0212 (FIFO), 0.0211 (LWNQ), 0.0211 (FGCA), 0.0212 (SRPT++), 0.0213 (SRPT+), 0.0215 (SRPT), 0.0212 (EDD), 0.0209 (STNV).

Note that it is fair to compare scheduling policies against each other when the releases in each case are either Deterministic and of the same rate, or Poisson and of the same rate. It is also fair to compare the mean cycle–times for different scheduling policies against each other, when the same Closed–Loop Release policy is used in each case. The reason is that, by Little's Law, for a fixed number of lots in the plant, the throughput rate is proportional to the mean cycle–time. However, when comparing different release policies against each other, while holding the scheduling policy fixed, such as WR(2975) against Deterministic Release of rate 0.0212, as we do in Fab 3′, or Closed–Loop Release with Deterministic Release, one should keep the throughput rate differences in mind.

Several pilot runs were made to determine the length of the transient period. It was determined that for a run of 3000 lots through an initially empty R&D line, statistics collected over the last 1000 lots were sufficient to capture the steady state performance under the Deterministic and Closed–Loop releases. On the other hand, a run of 6000 lots with statistics collected over the last 1000 lots was used for Poisson releases. For the Production Line, a run of 14000 lots was used for all release policies. Statistics were collected on the succeeding 4000 lots.

Tables 6 and 7 provide the averages over 20 simulation runs of the mean queueing time (MQT) and the standard deviations of the cycle–times (SDCT), for each combination of fab line and release policy tested. The queueing time is defined as the actual cycle–time minus the expected value of the total processing time.

Each of the 20 simulations was run with a different seed set. Each seed set is responsible for generating the streams of random numbers representing the times between machine failures, times to machine repairs, lot processing times, and times between lot releases, where this is appropriate. The seed sets were held fixed across the scheduling policies, the release policies, and the different fabrication lines. Similar to the split–plot design in an agriculture setting, this achieves the homogeneity of the test subjects across treatment levels, and thus enables meaningful fair comparisons of treatment levels.

For each release policy and each fabrication line, the simulations thus yield an n by p matrix of mean queueing times, and a similar n by p matrix of cycle–time standard deviations. Here, the value of $n = 20$ denotes the number of simulations performed for each scheduling policy, while p is the number of different scheduling policies tested.

6 Statistical analysis of simulation results

To evaluate the import of the numbers obtained in the simulations, and determine whether the superior performances of the FSMCT and FSVCT policies are significant, we resort to the method of Multivariate Analysis of Variance (MANOVA). We use the special case, the repeated measure design, where the comparisons of the treatment levels, i.e., different scheduling policies in the experiments, are carried out on a common subject, i.e., the seed set, so that the resulting comparisons are meaningful. First we test against the null hypothesis stating that all treatment levels (the scheduling policies) have the same effect on the subjects. If this hypothesis is rejected, a follow up pairwise comparison will then indicate which policy has significantly better overall performance.

Since, by inspection, the STNV Policy showed exceedingly bad performance consistently, we viewed it as an outlier, and excluded it from further statistical study, as well as from the plots in the Figures 3–7.

Four tests were used; the Lawley–Hotelling trace test, Wilks' likelihood–ratio test, the Pillai–Bartlett trace test, and Roy's maximum–root test. They are based on different sufficient statistics and are thus different in power and size [15]. All four tests can be shown to

be robust against the non–normality of the MANOVA model. However, statisticians suggest that the size of the Pillai–Bartlett test is the most robust, and that of Roy's test is the least robust [15].

All four tests yielded the same levels of significance shown in Table 8. Two values are cited in every entry. The first is intended for the mean cycle time, and the second for the standard deviation of the cycle time. The tests are significant at the 0.05 level for all release policies in all fabrication lines. This suggests that there exists at least a pair of scheduling policies that perform differently from each other.

Tukey's pairwise comparisons test, see [15], provides a way to locate the source of significance when the overall null hypothesis is rejected. It provides a set of homogeneous groups, which are groupings of scheduling policies so that the highest and lowest means do not differ by more than the shortest significant range for that group [16]. The best group for each combination of fab line and release policy is shown in Table 9.

7 Results of simulation experiments

The averages of the results obtained from the 20 simulations for each scenario tested are shown in Tables 6 and 7. Since the information contained there may be too much to digest, we provide some bottom line comparisons.

The best policy for the mean queueing time (averaged over the 20 simulations) was FSMCT, and the best policy for the standard deviation of cycle–times was FSVCT, in every combination of fabrication line and release policy, except three. For Fab 3′ under Deterministic Release FSVCT was very slightly better than FSMCT on MQT, while in Fabs 2 and 3 under Poisson Release FSMCT was better than FSVCT on SDCT, and FSVCT was second. Moreover, the second best policy for SDCT was always FSMCT, in every combination of fabrication line and release policy, except for the two cases cited above, where it was actually the best, and for the production line with Deterministic Release.

The statistical analysis reported in Table 9 shows that in *every* case, the recommended FSMCT policy was in the best group for mean cycle time, and the recommended FSVCT

was in the best group for standard deviation of cycle–time. No other policy comes close to possessing such behavior.

The magnitudes of the improvements obtained for the mean queueing time and standard deviation of cycle–time are reported in Figures 3–7. They show the percentage improvement of each scheduling policy over the baseline FIFO policy. The new policies FSMCT and FSVCT are highlighted in solid black. It should be noted (see Section 3.3) that under Deterministic releases, the two policies FSMCT and FSVCT coincide.

The results are unambiguous in supporting our earlier hypotheses that FSMCT reduces the mean cycle–time, and FSVCT reduces the standard deviation of cycle–time. The reduction in standard deviation of cycle–times is striking in all cases. The reductions in mean are large under Deterministic, Poisson and and Workload Regulation Releases. Under Closed–Loop Releases also our new policies FSMCT and FSVCT are best with respect to the mean cycle–time. However, the Closed–Loop Release policy has the effect of making the mean cycle–time relatively insensitive with respect to the scheduling policy, and thus the improvement in using the new policies is smaller.

Comparing the release policies, we see that the preferred choice is Workload Regulation Release. It has the lowest mean cycle–time, slightly lower than Deterministic, as reported in Wein [1]. This is easiest to see in Fab 3′ (see Table 6), which provides the straight head–to–head comparison of the Workload Regulation Release WR(2975) with Deterministic Release of rate 0.0212. However, what is more striking is that it has the lowest standard deviation of cycle–times, substantially better even than Deterministic Release. However, one should keep the throughput rate differences in mind when comparing different policies, as noted in Section 5.

One way to digest our conclusions is shown in Tables 10 and 11. In Fab 3, the choice of Workload Regulation Release over Deterministic Release reduces the mean queueing time of FIFO by 1.9%, and the standard deviation of cycle–time of FIFO by 28.1%. The choice of FSMCT instead of FIFO, *further* reduces the mean queueing time by 22.4%, and the standard deviation of cycle–time by 40.5%. The corresponding numbers for FSVCT instead

of FSMCT are 21.2% for mean queueing time, and 52.0%, respectively. Thus by adopting the WR-FSMCT release-scheduling pair, instead of the Deterministic-FIFO pair, we are able to reduce the mean queueing time from 1317.7 to 1003.0, and the standard deviation of cycle–time from 240.9 to 103.7. These are substantial improvements.

8 Conclusions

We have developed a new class of scheduling policies, called Fluctuation Smoothing (FS) Policies, for reducing the mean and standard deviation of cycle–times. These belong to the class of Least Slack policies. We have tested our policies on three models of Research and Development Fabrication Lines, and one model of a Production Line.

In all fab lines and for all release policies, the suggested FSMCT/FSVCT policies were best for reducing the mean queueing time and the standard deviation of cycle–time. The improvements obtainable through the use of the FSMCT and FSVCT scheduling policies are substantial for all release policies and fab lines tested, with the exception of the reduction in mean cycle–time under Closed–Loop Releases. The Closed–Loop Release policy renders the mean cycle–time relatively insensitive with respect to the scheduling policy, and hence the benefits of using our FSMCT policy are smaller.

Our bottom line recommendation is to use the FSMCT policy to reduce the mean cycle–time. It also does very well with respect to the standard deviation of the cycle–times.

With regard to the release policy, we recommend the use of Workload Regulation Release Policies. They provide a slightly lower mean cycle–time than Deterministic Release. However, the major advantage of Workload Regulation Releases is that they improve the standard deviation of cycle–time substantially.

Thus, we recommend the joint use of Workload Regulation (WR) Releases together with FSMCT for scheduling. The improvements in mean queueing time and standard deviation of cycle–time can be substantial. For example, in Fab 3, adopting the WR-FSMCT release-scheduling pair, instead of the Deterministic-FIFO pair, reduces the mean queueing time from 1317.7 to 1003.0, and the standard deviation of cycle–time from 240.9 to 103.7.

For future work. one could investigate more active versions of our Fluctuation Smoothing scheduling policies. For example, one could let the estimates ζ_i of remaining cycle–time be state dependent. As one possibility, they could be chosen to depend on just the failure or repair status of some critical machines. Or one could even attempt to expedite to bottlenecks.

Also, more work remains to be done on systems with multiple process flows, and on batching of lots.

Acknowledgments The authors are grateful to Sarah Hood of IBM for informative discussions on modeling a production line.

References

[1] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 115–130, August 1988.

[2] J. M. Harrison and L. M. Wein, "Scheduling networks of queues: Heavy traffic analysis of a two-station closed network," *Operations Research*, vol. 38, no. 6, pp. 1052–1064, 1990.

[3] C. R. Glassey and M. Resende, "Closed-loop job release control for VLSI circuit manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 36–46, February 1988.

[4] C. Lozinski and C. R. Glassey, "Bottleneck starvation indicators for shop floor control," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 147–153, November 1988.

[5] P. B. Chevalier and L. M. Wein, "Scheduling networks of queues: Heavy traffic analysis of a multi–station closed network," tech. rep., Sloan School of Management, M.I.T., Cambridge, MA, 1977. To appear in *Operations Research*.

[6] R. Uzsoy, C.-Y. Lee, and L. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry – Part I: System characteristics, performance evaluation and production planning," *IIE Transactions on Scheduling and Logistics*, vol. 24, pp. 47–61, 1992.

[7] R. Uzsoy, C.-Y. Lee, and L. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry – Part II: Shop floor control," Research Memorandum 92–14, Purdue University, West Lafayette, IN, 1993. Industrial Engineering. To appear in IIE Transactions on Scheduling and Logistics.

[8] P. R. Kumar, "Re–entrant lines," *Queueing Systems: Theory and Applications: Special Issue on Queueing Networks*, vol. 13, pp. 87–110, May 1993.

[9] D. J. Miller, "Simulation of a semiconductor manufacturing line," *Communications of the ACM*, vol. 33, no. 10, pp. 98–108, 1990.

[10] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*. Reading, MA: Addison-Wesley, 1967.

[11] S. H. Lu and P. R. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Transactions on Automatic Control*, vol. 36, pp. 1406–1416, December 1991.

[12] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY: Wiley–Interscience, 1975.

[13] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*. New York, NY: Wiley–Interscience, 1976.

[14] A. P. J. Vepsalainen and T. E. Morton, "Improving local priority rules with global lead–time estimates: A simulation study," *Journal of Manufacturing and Operations Management*, vol. 1, pp. 102–118, 1988.

[15] E. L. Lehmann, *Testing Statistical Hypotheses*. New York, NY: John Wiley & Sons, 1986.

[16] *SPSS-X User's Guide*. Chicago, IL: SPSS Inc., 3rd ed., 1988.