# Queueing Network Models in the Design and Analysis of Semiconductor Wafer Fabs

Sunil Kumar and P.R.Kumar

*Abstract*— We provide an introduction to the application of queueing network models to the design and analysis of semiconductor wafer fabs. We introduce the basic issues that confront the system manager and discuss a variety of queueing network based tools for addressing these issues. A representative collection of existing results in this area is also briefly surveyed.

*Keywords*—**Seminconductor wafer fabs, scheduling, queueing networks, performance evaluation**

## I. Introduction

In recent years there has been much progress in using queueing networks to model, analyze, and optimize the scheduling of semiconductor manufacturing plants. Indeed, over the past several decades, queueing theory has been repeatedly re-stimulated by a sequence of technological problems. The first wave of work in queueing theory was motivated by the problems of telephone exchanges; see Erlang [1]. The second wave was motivated by problems in job shops (Jackson [2]), and the third wave by the problems in performance evaluation of computer systems (Baskett, Chandy, Muntz, and Palacios [3]). Over the past dozen years, the focus on semiconductor manufacturing has led to a fourth wave of activity in queueing networks. This has led to many new developments including the discovery of instability in even simple queueing networks, stability proofs of several scheduling policies, the development of linear programming search procedures for Lyapunov functions, the fluid limit approach to stability, the development of ever sharper linear program performance bounds, the Brownian network approach to synthesis of scheduling policies, and the development of efficient scheduling policies for wafer fabs. In this paper we will present an account of some of these research results.

This paper does not attempt to be comprehensive in its efforts to survey existing results in this field. Rather it tends to serve as a tutorial introduction to one stream of research. Representative examples of research not explored in detail in this paper include the work of Connors, Fei-gin, and Yao [4], Gershwin [5], [6], Leachman *et. al.* [7], Dallery [6], [8], and Atherton and Dayhoff [9].

A natural way to model many manufacturing systems (and not just wafer fabs) is as a multiclass queueing network. In such a network, units of work (such as wafer lots) flow through the system along prescribed routes, requiring tasks (such as lithography) to be performed on them by resources (tools such as steppers). These units of work contend with each other for the attention of the resources and queue up in front of the resources to await processing when the resource is engaged in processing other units of work. The units of flow are differentiated by "class" and members of a different class may have different processing requirements at the same resource, as well as different routes through the network. The rest of this paper implicitly uses this generic model.

To set the problem in context, we begin by describing typical decisions that are taken in scheduling general manufacturing systems, and the various performance measures that system managers attempt to optimize. Then we discuss issues specific to scheduling semiconductor manufacturing systems. We discuss some commonly used sequencing rules and work release policies in a make-to-order setting. We illustrate how establishing even stability (that is, boundedness of average inventory) of such systems can be non-trivial. We then shift gears and discuss some analytical tools based on queueing network theory for performance evaluation and design of scheduling policies. We begin by describing a queueing network model called a reentrant line that is particularly suited for modeling wafer fabs. Using this model, we discuss two ways of establishing stability of scheduling policies: quadratic functionals and fluid models. We then discuss a way of estimating the performance of systems using quadratic functionals. Turning our attention to policy design, we discuss a heuristic approach validated by simulation as well as an approach based on Brownian approximations. Finally, we conclude by discussing several open research problems in this area.

## II. Scheduling

In the context of manufacturing systems, the term "scheduling" refers to the control of the flow of in-process material (commonly termed Work-in-process inventory or WIP) on the factory floor. Typically, the flow is controlled so as to achieve production targets set by the production planning and control department, while attempting to optimize some performance measures. The production planning and control department usually sets the production targets based on the number of outstanding customer orders (the "backlog"), or the level of the finished goods in-

Sunil Kumar is at the Graduate School of Business, Stanford University, Stanford, CA 94305-5015, email: skumar@stanford.edu

P. R. Kumar is at the Coordinated Science Laboratory and Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, email: prkumar@decision.csl.uiuc.edu

ventory required. These targets have to be met in a timely and efficient manner by the shop floor control (SFC) system. This system initiates production of new wafers and also tracks and controls the flow of work in process so as to achieve the production targets in the shortest possible time, while utilizing the plant and equipment and the workforce in the most efficient manner. These decisions implemented by the SFC system are broadly characterized as "scheduling."

Scheduling involves taking a variety of decisions regarding the flow of WIP on the shop floor. These decision are taken based on planning as well as the current status of the fab. For example, deciding when to release a new set of wafers into the fab will be based on both the production targets set for that month, and the number of wafers of that type currently waiting to complete processing in the fab. Typically, scheduling a fab includes the following.

• *Work Release:* Deciding the release of a new set of wafers to begin processing in the fab. This involves both a timing decision, i.e., when to release the new set of wafers, as well as a type choice decision, i.e., which one of the many types of wafers processed at the facility must be released next. The latter decision regarding the type of wafer released determines the *product-mix.*

• *Routing:* Deciding which one of the many tools capable of performing a particular processing step (usually grouped together at "stations") will be actually used to process a particular lot. Typically, these decisions are not planned in advance, but are taken dynamically when the wafer completes the previous processing step, based on which tools have failed, the workload on the tools, etc.

• *Sequencing:* Deciding when to work on a particular wafer at a particular tool. While in principle this could involve developing a detailed schedule for processing at that tool, usually, such detailed schedules are not used. Rather, rules are put in place for deciding which among the waiting wafers will be processed *next.* These rules are called "sequencing rules." An example of a sequencing rule would be First-Come-First-Serve (FCFS) which picks that wafer or lot of wafers that arrived first among those waiting for processing.

• *Lot Sizing:* Deciding the size of the sets that wafers travel together in between processing steps.

• *Batching:* Some "batch" tools process more than one wafer at a time. Deciding how many wafers to load at a given time into a batch tool is also one of the scheduling decisions.

• *Workforce and Preventive Maintenance Scheduling:* These decisions take place on a time-scale that is slower than the previous decisions and will not be discussed further in this paper. However, we note in passing that scheduling the workforce is particularly hard because of complications involving union regulations and other human resources issues.

The scheduling decisions outlined above have to be taken so as to optimize the trade-offs between various performance measures of interest. In order to utilize invested capital properly, one must utilize plant and equipment efficiently. At the same time, one must not overload the plant with excessive work-in-process inventories. It is usually required to start filling as much of the plant's backlog as possible in a given period. At the same time, one must fill these orders in the shortest possible time. Thus, there are many dimensions of performance of a fab scheduling policy. First, let us describe some of the metrics by which the performance of a policy is measured. Then we will attempt to describe tools for estimating these performance measures, as well as tools for policy design that attempt to optimize one or more dimensions of performance and thus help design policies that schedule the fab efficiently.

Performance metrics that are typically used in evaluating the operational performance of a wafer fab include the following.

• *Throughput Rate:* Also called just "throughput," this is the number of completed wafers exiting the fab in a given period, measured for example, in wafers per day. If the line yield (the fraction of usable product obtained at the end of the process) is 100%, then this is also the rate of *wafer starts* into the fab. In general, throughput rates are defined separately for each type of wafer processed in the fab. If there is more than one wafer type made in the fab, then the throughput rate is a vector with each component representing the throughput for the respective type.

• *Throughput Capacity:* This is the maximum sustainable throughput rate of a fab operating under a given scheduling policy. This is a fundamental limit to the achievable performance of the fab, and is determined by the throughput capacity, i.e., the maximum sustainable throughput rate, of each processing station considered in isolation. The throughput capacity is a function of *processing times, setups and lot size, the numbers of tools, tool failures, yields, batching rules,* etc. As we will show in a later section, surprisingly, the throughput capacity of a fab is not independent of the choice of scheduling policy.

• *Utilization:* The throughput rate in a fab cannot exceed the throughput capacity of the constraining station, the bottleneck. As a consequence other stations are *idle* for a fraction of the time. The fraction of the time a station is *not* idle is called the utilization of the station. In order to fully exploit the capital invested in obtaining the tools at that station, it is desirable to minimize the idleness of the station.

• *Throughput Time, Lead Time or Cycle Time:* All these terms are used to denote the total time taken by a wafer from when it is released into the fab to when it emerges from the fab as a completed wafer. This measures the responsiveness of the fab as well as its ability to achieve on-time delivery. Long throughput times also increase the time for which the wafer is exposed to potential contamination in the fab, and hence can result in lower yield.

• *Work In Process (WIP) inventories:* WIP inventory is the number of wafers which are still in the fab at various stages of processing. WIP inventory represents working capital tied up in the fab. Large WIP inventories also result in slower detection of quality problems, and general sluggishness of the fab.

## III. Scheduling Semiconductor Fabs

In this section, we discuss a representative set of scheduling rules which have been developed in particular for scheduling semiconductor fabs. For concreteness, we will concentrate on a make-to-order setting where the fab is run to fill outstanding customer orders and not to build up finished goods inventory. We shall begin by introducing various rules that have been used for general job shop type manufacturing systems. Then we shall point out the difficulties with using these policies in a semiconductor fab, and thus motivate the need for designing policies that are specialized for semiconductor fabs. We begin by describing scheduling rules which have been used for many years in job shop manufacturing settings.

### A. Common Sequencing Rules

There are a wide variety of sequencing rules which have been developed for general job shop type manufacturing systems [10]. Most of these rules have been developed using heuristics which attempt to control either (a) the configuration of the WIP inventory, and/or (b) the material flows within the manufacturing shop floor, as a way of attempting to optimize the trade-off between throughput, lead time and WIP inventory on the shop. As a brief introduction to the vast array of available rules, we present a short list of representative sequencing rules and the rationale behind each of them.

To recall, sequencing rules are policies which decide which of the wafer lots waiting for processing at a tool is to be processed next at that tool. The one all of us understand and know about is the *First In First Out (FIFO)* rule. This rule picks that lot which has waited at the tool the longest for service. Another popular policy is the *Shortest Processing Time* rule. This rule picks that wafer lot which has the least amount of processing time requirement from that tool. The rationale is that one wishes to get the short jobs out to the next processing step as quickly as possible. Alternatively, one can think of getting lots out of the *entire system* as soon as possible. This is motivated by the desire to reduce the throughput time of the jobs. One way to try and achieve this is to choose a scheduling rule which picks that lot for processing which has the least amount of total processing left before it exits the entire system. This rule is called the *Shortest Remaining Processing Time* rule. On the other hand, one can argue that at any station, attention must be given to that lot that requires the maximum amount of work from the tool, before attending to shorter jobs. This results in the *Longest Processing Time* rule. Another set of flow control based sequencing rules is the set of *Least Slack* policies. These policies take into account the *due dates* of the wafer lots, and give priority to those lots that have the least amount of *slack*, i.e., the one that are closest to their due dates or are the most past due. As one can imagine, the number of such heuristics is tremendous. Rather than attempting an exhaustive survey, we will conclude with one more interesting heuristic sequencing rule. The rules we have described above all attempt to regulate the flow of wafer lots on the fab floor. Alternatively, we can think of policies that attempt to regulate the inventory levels at each of the stations. One heuristic that does this is the *Least Work Next Queue* rule, where priority is given to the wafer lot that, on completion of processing, will join the queue of waiting lots at the next processing step that has the least amount of work waiting to be done. Thus, this rule attempts to regulate WIP inventories at the next, downstream station and provide work for that station that is most likely to be starved.

Some sequencing rules are designed to mitigate the impact of set-ups. One way to minimize the impact of set-ups is to serve all the wafer lots that can be processed using the current set-up, until no more such lots are available for processing at the tool, before switching to processing another type of wafer and thus having to do a set-up. This is the *serve to exhaustion* or *clearing* rule.

Another set of commonly used scheduling rules worth discussing are the batching rules. The batching decision involves deciding when and how many wafers to load into a batch tool for simultaneous processing. The trade-off is whether to start as soon as possible and possibly run the batch tool with fewer wafers than the tool is capable of handling simultaneously, or to wait until enough wafers have accumulated at the tool to fully utilize the capacity of the tool, at the risk of increasing the delay experienced by the already present wafers. One commonly used batching rule is the *limited look-ahead* rule, where one waits to see if there are any wafers arriving in the near future (up to a limited time horizon) before loading and starting up the batch tool.

### B. Common work release policies

Scheduling policies also attempt to regulate the flow of work onto the factory floor in an attempt to optimize the trade-off between throughput rate and cycle time. In this subsection we discuss some common release policies to illustrate the various issues that must be grappled with in designing such policies.

In designing release policies, one must try and attain the throughput rate required to make sure that the backlog of customer orders does not grow without bound, while still maintaining a small amount of WIP in the fab and keeping the mean cycle time small. One can just release work into the system as customer orders arrive and thus build up WIP on the fab floor. Arguably, it is better to keep the inventories on paper, i.e., as a pending order waiting to be released onto the fab floor than as WIP in the fab. Then these pending orders can be released along with the required raw material (in this case, a raw wafer) according to some mechanism which improves the performance of the fab.

One common release control mechanism used in general job shops is *deterministic release.* Here the orders are released onto the shop floor only at periodic intervals. This has the advantage of removing one potential source of variability from the system. This is an example of a release policy that attempts to regulate flows in the system. One could also conceive of release policies that attempt to reg-

ulate WIP on the fab floor. One such policy is the CON-WIP[11] policy, also known as the Closed Loop release policy [12]. CONWIP (which stands for CONstant WIP) explicitly controls the amount of WIP inventory on floor. It maintains the level of total WIP constant. This policy is usually implemented by releasing a new wafer lot onto the floor only when a completed lot of the same type leaves the floor. One attempts to match the throughput rate required to achieve the production targets by increasing the constant level of WIP being maintained. Increasing the WIP usually increases the throughput rate, but it also increases the cycle time as well. Thus a balance needs to be struck between the allowed WIP level and the target throughput rate.

An extension of this policy is to explicitly maintain the level of WIP constant at every processing step. One way to do this is to allow a transfer of a lot from one processing step to its succeeding processing step only when the succeeding step completes a transfer. That is, the downstream step *pulls* work in from the upstream step as it completes and delivers its own work further downstream. This method of WIP control was popularized by the Japanese automobile industry and is called the *Kanban* system. Of course one can conceive of variants that lie somewhere between CONWIP and Kanban. See [8] for a unified treatment of such pull mechanisms.

As is evident from the above, there is a plethora of suggested sequencing rules and work-release policies. One of the motivations behind developing queueing networks is to allow us to make a natural choice of policy.

### C. Scheduling difficulties particular to wafer fabs

Wafer fabs have certain characteristics that make them particularly hard to schedule. The most important of these is the *complexity of process flow*. The production of wafers involves several hundred processing steps. These steps consist of similar operations that are repeated for each layer. For example, the "expose" step in photolithography is repeated for each of the 15 or so layers that form a VLSI chip. The economic necessity of reducing capital investment, as well as some technological requirements such as maintaining alignment by using the same optics, force sharing of equipment between lots that differ in the layer being processed. That is, the same stepper may be used to expose wafers at different layers. As a consequence, the flow of wafers in a fab is a complex *reentrant line*. A representative reentrant line is shown in Figure 1, and a representative process sub-sequence is shown in Figure 2. It is seen that wafers repeatedly return to the same station for the processing of subsequent layers. In the standard manufacturing process spectrum, the reentrant line topology of wafer fabs places them somewhere in between classical line flow manufacturing systems and classical job shops.

The other characteristic of wafer fabs that makes them hard to schedule [13] is the *diversity of equipment*. The equipment (or "tools," as they are often called) vary widely. Some of the tools process wafers one at a time. Such "serial processing" could involve significant set-up and change-
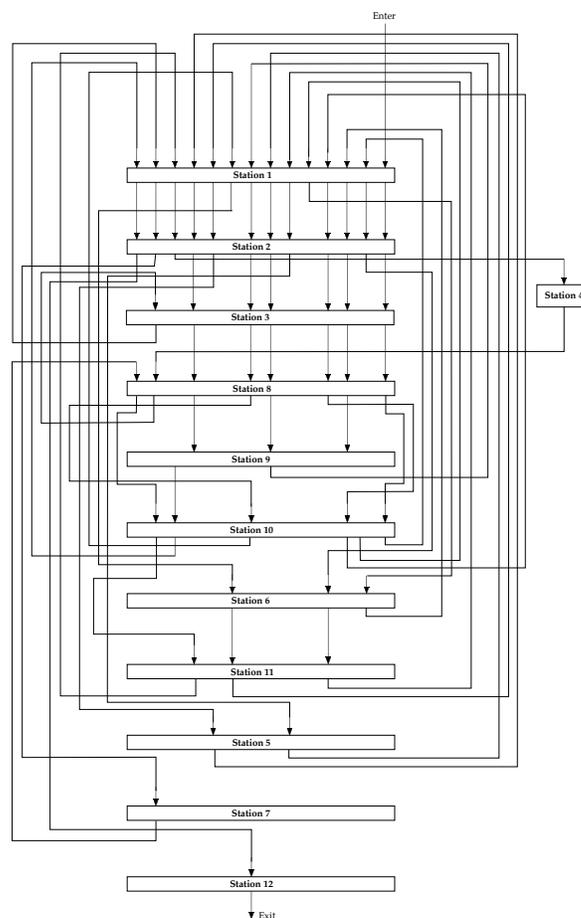


Fig. 1. A Representative Reentrant Line

over times when switching from one wafer to another. An example of such a tool is the stepper which exposes one wafer at a time. Other tools called "batch" tools process a batch of wafers at the same time. An example of a batch tool is the drive furnace in Figure 2. Batch tools also have set-up and change-over times. Finally, *stochastic variability* especially in the form of random machine failures, random order arrivals in a make-to-order environment, and variable operators' processing times, is an additional complicating factor.



Fig. 2. A Typical Process Subsequence

Now, we will motivate the need for designing scheduling policies especially for wafer fab scheduling, using a highly idealized reentrant segment of a fab. This example is mo-

Fig. 5.   Reentrant Line Model



Fig. 3.  Stylized Example

In the spirit of the Shortest Remaining Processing time rule, processing steps 4 and 3 are given priority at tools 1 and 2 respectively, since they correspond to exit steps. The release policy is deterministic, and wafers lots of both type A and type B are released into the system periodically at 75 minute intervals. The total WIP in the system versus time is plotted for a simulation run in Figure 4.

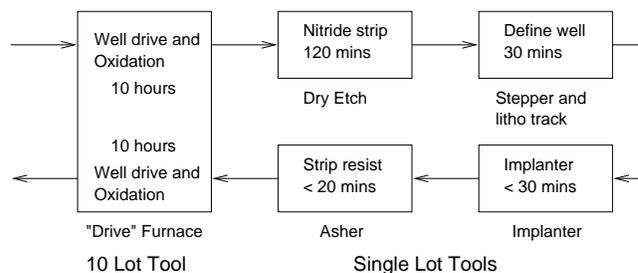As we can see from Figure 4, the WIP inventory increases without bound. This is definitely not what could be predicted from a naive analysis of the situation presented here. For example, each pair of wafers of type A and B entering the system brings with it 70 minutes worth of work for tool 1 (since step 1 takes one hour and step 2 takes 10 minutes) and since wafer pairs come in every 75 minutes, one expects that tool 1 will be capable of handling this work and would be busy about $\frac{70}{75}$ or 93.3% of the time. But this is not the case. The reason for this bizarre behavior is the highly reentrant nature of the flow, combined with a poor



Fig. 4.  WIP trajectory in Example

choice of scheduling policies.  The priority policy causes alternate blocking and starvation of the tools, resulting in WIP increasing without bound because the tools lose too large a fraction of their time being starved for work to be able to complete the workload imposed on them.  Thus, this is an example of how throughput capacity of the fab may depend on the scheduling policy employed.

Although this example is in a very simple setting, its moral carries over to real fabs – a naive choice of scheduling policies combined with reentrant line flow could result in very non-intuitive and undesirable behavior. This motivates the need for better policy design for scheduling wafer fabs, that take the special features of the wafer fab into account.

## IV. The Basic Queueing Network Model: Reentrant Lines

We now shift gears and begin discussing analytical tools for addressing the practical issues raised in the previous section. We begin by describing a queueing network model that is particularly relevant to seminconductor wafer fabs, called reentrant lines [16]. Consider the system shown in Figure 5. There are $L$ buffers, $b_1, b_2, \ldots, b_L$, and several machines. We shall denote by $\sigma(i)$, the machine serving buffer $b_i$. We allow $\sigma(i) = \sigma(j)$ even if $i \neq j$. Parts arrive as a Poisson process of rate $\lambda$ to buffer $b_1$. They move from $b_i$ to $b_{i+1}$ after obtaining service at $b_i$. After completing service at $b_L$, they leave the system. Parts at buffer $b_i$ require a processing time which is exponentially distributed with mean $\frac{1}{\mu_i}$. As is usual, we assume that the interarrival times and service times are independent.

Let $x_i(t)$ denote the number of parts in buffer $b_i$ at time $t$. Since $t$ is a continuous time variable, this is a continuous time system, which we take to be right continuous with left limits. However we prefer to study a discrete time system, obtained by sampling as follows. If a part in buffer $b_i$ is not being processed by the corresponding machine, we shall suppose that there is a fictitious or virtual part that is being processed. We sample the system at all arrival times, and at all real or virtual service completion times. The rates at which these events occur are $\lambda$ and $\mu_i$, respectively. We shall now assume (by rescaling time, if necessary) that,

$$\lambda + \sum_{i=1}^{L} \mu_i = 1.$$

Let $\{\tau_n\}$ be the sequence of sampling times, obtained

above, with $\tau_0 := 0$. This procedure (called "uniformization," see [17]) yields a discrete-time stationary Markov chain whose steady state distribution is the same as the underlying state process $x(t) = [x_1(t), \ldots, x_L(t)]'$, if this process were a continuous time Markov chain[1].

We shall suppose that the scheduling policy does not change the buffer that it is working on during the time interval $[\tau_n, \tau_{n+1})$. We shall call such scheduling polices as "non-interruptive," and restrict attention to them.

Define the random variable $w_i(\tau_n) := 1$ if the scheduling policy is actually working on a real part in buffer $b_i$ in the time interval $[\tau_n, \tau_{n+1})$, and $:= 0$ otherwise.

If $e_i$ is the $i$-th coordinate vector, then the discrete time system $x(\tau_n) = (x_1(\tau_n), \ldots, x_L(\tau_n))$ is described by

$$
x(\tau_{n+1}) = \begin{cases}
x(\tau_n) + e_1 & \text{w.p. } \lambda, \\
x(\tau_n) - e_i + e_{i+1} & \text{w.p. } w_i(\tau_n)\mu_i \\
& \text{for } 1 \le i \le L-1, \\
x(\tau_n) - e_L & \text{w.p. } w_L(\tau_n)\mu_L, \\
x(\tau_n) & \text{otherwise.}
\end{cases} \quad (1)
$$

Let us now restrict our attention to scheduling policies that are *non-idling*. These are policies where a machine is not allowed to stay idle if any one of its buffers is non-empty. Thus,

$$
\sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n) = 1 \text{ whenever } \sum_{\{i:\sigma(i)=\sigma\}} x_i(\tau_n) \ge 1. \quad (2)
$$

In later sections we will consider buffer priority policies. We model such policies using a permutation $\theta : \{1, 2 \ldots, L\} \to \{1, 2 \ldots, L\}$ such that if $\sigma(i) = \sigma(j)$ and $\theta(j) < \theta(i)$, then buffer $b_j$ is preferred to $b_i$. That is, if $x_j(\tau_n) \ge 1$, one has $w_i(\tau_n) = 0$. Hence, in (9), one can drop such $w_i(\tau_n)$'s. In particular we will consider one particular buffer priority policy called *Last Buffer First Serve (LBFS)* which is identical to the Shortest Remaining Process Time policy for reentrant lines, and is specified by the ordering $\theta(i) = L - i$.

The model described here can be extended in many ways. It can be adapted to handle the work release policy CONWIP by simply treating the first buffer $b_1$ as if it were fed by the last buffer $b_L$. To be more concrete, the analog of (1) becomes

$$
x(\tau_{n+1}) = \begin{cases}
x(\tau_n) - e_i + e_{i+1} & \text{w.p. } w_i(\tau_n)\mu_i \\
& \text{for } 1 \le i \le L-1, \\
x(\tau_n) - e_L + e_1 & \text{w.p. } w_L(\tau_n)\mu_L, \\
x(\tau_n) & \text{otherwise.}
\end{cases} \quad (3)
$$

It can also be adapted to handle machine or tool failures in the following indirect way. At every machine one imagines a fictitious highest priority class such that whenever a customer is present in that it preempts the machine from working on any of the real classes and this simulates a failure. This class can be fed in a CONWIP fashion by a single customer to achieve the desired Mean-time-between-failures and Mean-time-to-repair statistics. Finally the

[1]However we have not so far assumed that the scheduling policy is stationary, and so the state process may not be a Markov chain.

model can handle reentrant lines that process multiple part types (such as the example of Figure 3) and probabilistic routing. The reader is referred to [23] for details.

## V. STABILITY

Through the work of Jackson [2], Baskett, Chandy, Muntz and Palacios [3], and Kelly [18], explicit solutions have been determined for the steady state distributions of certain queueing networks. Such networks are broadly referred to as "product form" queueing networks, due to the multiplicative form of their steady state probability distribution. Unfortunately, if the service time distributions for the parts in the buffers at the same machine are different (there are some minor exceptions, unimportant here), or if the scheduling policy gives priority according to the buffers, then the resulting networks generally fall outside the above class. As was evident from the example of the previous section, even the issue of stability of such non-product form networks is non-trivial to resolve in general.

### A. Stability using Quadratic Functionals

We will now describe a method for establishing the stability of stochastic queueing networks, defined for our purposes here as the existence of an unique stationary distribution for the underlying Markov chain of the system, by simply computing the value of a linear program and checking whether it is 1 or 0. The results in this subsection and the next are drawn from [19]. We will now examine whether all non-idling policies yield a stable system, i.e., if the system is stable for all choices of $\{w_i(\tau_n)\}$, subject only to the requirement that they depend only on the past, and satisfy (2).

Consider a quadratic functional, $x^T(\tau_n)Qx(\tau_n)$, where

$$
Q = Q^T.
$$

Let $\mathcal{F}_{\tau_n}$ be the past $\sigma$-algebra. Suppose that we can find a $Q$ such that

$$
E[x^T(\tau_{n+1})Qx(\tau_{n+1}) \mid \mathcal{F}_{\tau_n}] \le x^T(\tau_n)Qx(\tau_n) - \gamma|x(\tau_n)| + c, \quad (4)
$$

where $\gamma > 0$, $|x| := \sum x_i =$ number of parts in the system, and $c$ is some constant. (Note that the conditional expectation above exists, since $x(\tau_n)$ can grow only linearly with $n$). Then, we can take unconditional expectations to obtain,

$$
E(x^T(\tau_{n+1})Qx(\tau_{n+1})) \le E(x^T(\tau_n)Qx(\tau_n)) - \gamma E|x(\tau_n)| + c.
$$

By summing, and dividing by $N$, we obtain

$$
\frac{\gamma}{N} \sum_{n=0}^{N-1} E|x(\tau_n)| \le \frac{x^T(0)Qx(0)}{N} - \frac{E(x^T(\tau_N)Qx(\tau_N))}{N} + c. \quad (5)
$$

Suppose now that $Q$ satisfies the condition,

$$
x^T(\tau_N)Qx(\tau_N) \ge 0 \text{ for all } N. \quad (6)
$$

Then, (5) yields

$$\frac{\gamma}{N}\sum_{n=0}^{N-1} E|x(\tau_n)| \leq \frac{x^T(0)Qx(0)}{N} + c \text{ for all } N. \qquad (7)$$

Since the average of the mean number of parts in the system is then bounded, we can say that the system is "stable-in-the-mean." For Markov chains, stability-in-the-mean implies positive recurrence.

Now we turn to the crucial issue of finding a symmetric matrix $Q$ for which (4) holds. From the probabilities of the various transitions given in (1), it is easy to compute $E[x^T(\tau_{n+1})Qx(\tau_{n+1}) \mid \mathcal{F}_{\tau_n}]$. Simple calculations show that (4) is equivalent to finding a $Q = Q^T$ such that,

$$2\lambda e_1^T Qx(\tau_n) + \lambda e_1^T Qe_1 + 2\sum_{i=1}^{L-1}\mu_i w_i(\tau_n)(e_{i+1}-e_i)^T Qx(\tau_n)$$

$$+\sum_{i=1}^{L-1}\mu_i w_i(\tau_n)(e_{i+1}-e_i)^T Q(e_{i+1}-e_i)$$
$$-2\mu_L w_L(\tau_n)e_L^T Qx(\tau_n) + \mu_L w_L(\tau_n)e_L^T Qe_L$$
$$\leq -\gamma|x(\tau_n)| + c \text{ for some } \gamma > 0 \text{ and } c.$$

Bounding all the terms not involving $x(\tau_n)$ on the left hand side above by a constant, we see that (4) is equivalent to

$$2\lambda e_1^T Qx(\tau_n) + 2\sum_{i=1}^{L-1}\mu_i w_i(\tau_n)(e_{i+1}-e_i)^T Qx(\tau_n)$$

$$-2\mu_L w_L(\tau_n)e_L^T Qx(\tau_n) \leq -\gamma|x(\tau_n)| \text{ for some } \gamma > 0. \qquad (8)$$

Now we observe that both the left and right hand sides above are linear in $x(\tau_n)$. Hence (8) (and therefore (4)) holds if the coefficient of every component of $x_j(\tau_n)$ is less than $-\gamma$, whenever $x_j(\tau_n) \geq 1$ (note that $x_j(\tau_n)$ is integral), i.e., for $j = 1, \ldots, L$,

$$\lambda q_{1j} + \sum_{i=1}^{L}\mu_i(q_{i+1,j}-q_{ij})w_i(\tau_n) \leq -\gamma \text{ if } x_j(\tau_n) \geq 1. \qquad (9)$$

(Above we have taken $q_{L+1,j} := 0$). Thus we seek $q_{ij} = q_{ji}$ such that (9) holds.

*Theorem 1* (Kumar and Meyn) Consider the linear program:

$$\max \gamma$$

subject to

$$0 \leq \gamma \leq 1$$
$$q_{ij} = q_{ji} \text{ for all } i, j$$
$$q_{L+1,j} = 0 \text{ for all } j$$
$$\lambda q_{1j} + r_j + \sum_{\{\sigma:\sigma\neq\sigma(j)\}} s_{\sigma j} \leq -\gamma \text{ for all } j$$
$$r_j \geq \mu_i(q_{i+1,j}-q_{ij})$$
$$\text{for all } i \text{ with } \sigma(i) = \sigma(j), \text{ and all } j$$
$$s_{\sigma j} \geq \mu_i(q_{i+1,j}-q_{ij})$$
$$\text{for all } i \text{ with } \sigma(i) = \sigma \neq \sigma(j), \text{ and all } j, \sigma$$
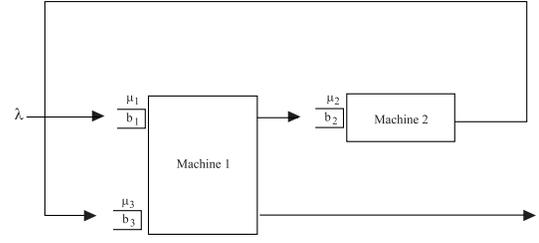$$s_{\sigma j} \geq 0 \text{ for all } \sigma, j.$$



Fig. 6. Reentrant Line Example

*i)* Suppose the value of the LP is 1, and suppose $Q$ is an optimal solution. If an optimal $Q$ is copositive[2], then *all* non-idling scheduling policies are stable-in-the-mean. Also, for all non-idling stationary scheduling policies, the system has a geometrically converging exponential moment.

*ii)* Consider the same linear program with the additional constraints

$$q_{ij} \geq 0 \text{ for all } i, j.$$

If the value of this LP is 1, then since every optimal $Q$ is nonnegative, the conclusions of (i) above hold without the need for a separate copositivity check of $Q$.

*iii)* If the value of the first LP is 0, then no conclusion can be drawn.

Example 1

Consider the system as in Figure 6, with $\mu_1 = \mu_3 = 1$, and $\mu_2 = 2$. Also assume that the arrival rate $\lambda = 0.4$. Then, we have

$$\rho_1 := \frac{2\lambda}{\mu_1} < 1, \text{ and } \rho_2 := \frac{\lambda}{\mu_2} < 1,$$

and the system is stable under all non-idling policies since the LP test above is satisfied with a positive $Q$.

*B. Stability via Fluid Models*

In this section we explore an alternate way of establishing the stability of the reentrant lines via *fluid models*, proposed by Dai [20]. In particular, we once again consider the reentrant line of Figure 6, now operating under the LBFS policy. Fluid models are continuous, deterministic approximations to the dynamics of the original discrete, stochastic system. The fluid model treats the contents of each buffer as if it were infinitely divisible fluid and each server as if it were a pump capable of infinitely divisible effort.

We can write the fluid model of the system of Figure 6 operating under the LBFS policy as follows. Let $\bar{z}_i(t)$[3], $i = 1, 2, 3$ denote the fluid levels of the three buffers and $\bar{T}_i(t)$, $i = 1, 2, 3$ denote the cumulative amount of time spent by the corresponding server on each of the buffer. Let $\dot{f}(t)$ denote the time derivative of function $f(.)$ at time $t$. It can be shown that the above quantities can be defined as the sequence of a stochastic solutions, and are

---

[2]$Q$ is copositive if $x'Qx \geq 0$ for any vector $x \geq 0$.
[3]The bar notation $\bar{z}$ is used to distinguish the quantity in the fluid model from the quantity in the original stochastic system.

non-negative, Lipschitz continuous functions, with $T_i(.)$, $i = 1, 2, 3$ non-decreasing, that satisfy the following equations almost everywhere. The reader is referred to Dai [20], and Dai and Weiss [21] for details.

$$\dot{\bar{z}}_1(t) = \lambda t - \mu_1 \dot{\bar{T}}_1(t) \tag{10}$$

$$\dot{\bar{z}}_2(t) = \mu_1 \dot{\bar{T}}_1(t) - \mu_2 \dot{\bar{T}}_2(t) \tag{11}$$

$$\dot{\bar{z}}_3(t) = \mu_2 \dot{\bar{T}}_2(t) - \mu_3 \dot{\bar{T}}_2(t) \tag{12}$$

$$0 = \int_0^t (\bar{z}_1(s) + \bar{z}_3(s)) \, d(s - (T_1(s) + T_3(s))) \tag{13}$$

$$0 = \int_0^t \bar{z}_2(s) \, d(s - T_2(s)) \tag{14}$$

$$0 = \int_0^t \bar{z}_3(s) \, dT_1(s) \tag{15}$$

The following result, due to Dai [20], helps us establish the stability of the reentrant line of Figure 6, operating under the LBFS policy.

*Theorem 2* (Dai) If there exists a time, $T$, depending only on $\bar{z}_i(0)$, $i = 1, 2, 3$ such that all solutions $\bar{z}_i(t)$, $i = 1, 2, 3$ to (10-15) satisfy

$$\sum_{i=1}^3 \bar{z}_i(t) \equiv 0 \text{ for all } t \geq T, \tag{16}$$

i.e., the system empties out in finite time under all solutions of the fluid model, the reentrant line operating under the LBFS policy is stable in the sense that the underlying Markov chain is positive recurrent.

We now need to establish (16). We will not show the analysis in detail. The reader is referred to Dai and Weiss [21] or Kumar and Kumar [22], where the result is established in general for all reentrant lines. Figure 7 shows a solution to (10-15) when $\lambda = 0.4$, $\mu_2 = 2$, $\mu_1 = \mu_3 = 1$, and $\bar{z}_i(0) = 1$, $i = 1, 2, 3$. Initially, fluid buffer $b_2$ drains. During this period the fluid level in buffer $b_3$ increases because rate of flow out of buffer $b_2$ and into buffer $b_3$ (equal to $\mu_2$) exceeds the rate at which buffer $b_3$ can drain fluid (which equals $\mu_3$). Once buffer $b_2$ is empty, buffer $b_3$ drains (at rate $\mu_3$) since there isn't any inflow. During the period when both buffer $b_1$ and buffer $b_3$ are positive, buffer $b_1$ receives no attention from the server because of the LBFS policy. Thus the level in buffer $b_1$ keeps growing during this period due to external arrivals. Once buffers $b_2$ and $b_3$ are drained, buffer $b_1$ pumps fluid out at the rate $\mu_1/2 = 0.5$ since the server is forced to devote equal time to both buffers $b_1$ and $b_3$. Since this outflow rate is larger the inflow rate $\lambda = 0.4$, buffer $b_1$ also drains eventually after which (16) is satisfied. (16) can be similarly established for all other initial conditions in a similar fashion, thus establishing the stability of the system.

### C. Work Release Policies and Stability

Consider a make-to-order fab that is operating under the CONWIP work release policy. In this case customer orders
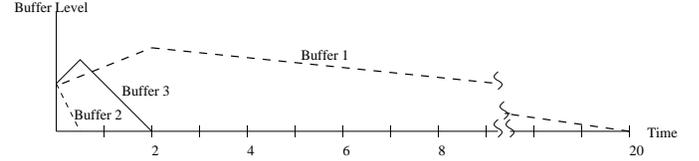


Fig. 7. Fluid Model Behavior of System under LBFS

are arriving in a random fashion to the fab, but the manager only releases an order into the fab floor when a completed order departs from the system. In this case, since the WIP on the fab floor remains constant, the reader may be tempted to believe that stability is not an issue. Yet, on closer inspection, one realizes that unless the fab is processing orders at a sufficiently high throughput rate, the backlog of customer orders not yet released to the fab floor will grow without bound. Of course the manager can try to increase the throughput rate by increasing the constant WIP level. But in the spirit of Figure 3, it is not necessarily true that that scheduling policies employed in the fab realize throughput rates sufficiently close to the fab's throughput capacity at any WIP level. Thus, one can define a notion analogous to stability under the CONWIP release policy, called *efficiency*, which says that when the constant WIP level increases without bound, the throughput rate achieved by the fab operating approaches the throughput capacity of the fab. See [23], [24] for details of how the analyses presented in this section can be adapted to establish efficiency of fabs operating under CONWIP work release policies.

A brief note of comparison of the two methods described here is in order. In the opinion of the authors the quadratic functional method is less problem specific than the fluid model approach. That is, less work is required in applying to a particular context. The fluid model approach on the other hand needs some problem specific analysis regarding emptying times for the fluid model. However, insights about the specific problem can be more easily incorporated in the fluid analysis. Either method may fail to give a conclusive answer: if the tests fail (we do not get $\gamma = 1$ in the quadratic functional approach and we cannot establish that the fluid model empties in finite time), it does not mean that the system is unstable.

### VI. PERFORMANCE BOUNDS

We now examine the issue of quantifying the performance of a system. The results in this section and the next are mainly drawn from [23]. Some sharpening of the results are from [25].

Suppose now that, under a particular stationary non-idling scheduling policy, the system is stable, i.e., positive recurrent. Suppose moreover that the system has a *finite first moment in steady-state*, i.e.,

$$E|x(\tau_n)| < +\infty, \tag{17}$$

in steady state. Then, it can be shown that,

$$E[x^T(\tau_{n+1}) Q x(\tau_{n+1}) - x^T(\tau_n) Q x(\tau_n)] = 0, \tag{18}$$

for *every* matrix $Q$. This is particularly easy to see if

$$E|x(\tau_n)|^2 < +\infty$$

in steady state, but the result (18) also holds when just (17) holds, as shown in [19].

Relation (18) is equivalent to,

$$E[x_i(\tau_{n+1})x_j(\tau_{n+1}) - x_i(\tau_n)x_j(\tau_n)] = 0 \text{ for all } i, j. \quad (19)$$

Note that this is a set of $\frac{L(L+1)}{2}$ equations.

To compute the left hand side of (19), we begin by computing the conditional expectation

$$E[x_i(\tau_{n+1})x_j(\tau_{n+1}) - x_i(\tau_n)x_j(\tau_n) \mid \mathcal{F}_{\tau_n}], \quad (20)$$

using the transition probabilities (1). There are several cases. Consider, for example, $i = j = 1$:

$$x_1^2(\tau_{n+1}) = \begin{cases} (x_1(\tau_n) + 1)^2 & \text{w.p. } \lambda, \\ (x_1(\tau_n) - 1)^2 & \text{w.p. } w_1(\tau_n)\mu_1, \\ x_1^2(\tau_n) & \text{otherwise.} \end{cases} \quad (21)$$

For $i = 1$ and $j = 2$,

$$x_1(\tau_{n+1})x_2(\tau_{n+1}) = \begin{cases} (x_1(\tau_n) + 1)x_2(\tau_n) \text{ w.p. } \lambda, \\ (x_1(\tau_n) - 1)(x_2(\tau_n) + 1) \\ \quad \text{w.p. } w_1(\tau_n)\mu_1, \\ x_1(\tau_n)(x_2(\tau_n) - 1) \\ \quad \text{w.p. } w_2(\tau_n)\mu_2. \end{cases}$$

It is a simple matter to write the other cases down.

We illustrate the consequence of (19), for $i = j = 1$. From (21), we see that (20) evaluates to

$$\begin{aligned} E[x_1^2(\tau_{n+1}) &- x_1^2(\tau_n) \mid \mathcal{F}_{\tau_n}] \\ &= \lambda(2x_1(\tau_n) + 1) - \mu_1 w_1(\tau_n)(2x_1(\tau_n) - 1. \quad (22) \end{aligned}$$

Let us define

$$z_{ij} := E[w_i(\tau_n)x_j(\tau_n)]. \quad (23)$$

Taking unconditional expectations in (22), we see that

$$\begin{aligned} E[x_1^2(\tau_{n+1}) &- x_1^2(\tau_n)] \\ &= 2\lambda E[x_1(\tau_n)] + \lambda - 2\mu_1 z_{11} + \mu_1 E[w_1(\tau_n)]. \quad (24) \end{aligned}$$

Now recall that since our scheduling policy is non-idling, we have the condition (2). Hence, $x_j(\tau_n) = \sum_{\{i:\sigma(i)=\sigma(j)\}} w_i(\tau_n)x_j(\tau_n)$, and so

$$E[x_j(\tau_n)] = \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij}. \quad (25)$$

Also, since the system is stable, every buffer $b_i$ is worked on for a proportion of time that is exactly enough to meet the work arriving for $b_i$. Hence,

$$E(w_i(\tau_n)) = \frac{\lambda}{\mu_i}. \quad (26)$$

Substituting (25) and (26) in (24), we see that

$$E[x_1^2(\tau_{n+1}) - x_1^2(\tau_n)] = 2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11}.$$

Hence, from (19), for $i = j = 1$, we obtain the equality,

$$2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11} = 0.$$

Similarly, by considering all the other cases for $(i, j)$, one can obtain $L(L + 1)/2$ equality constraints in the $L^2$ variables $\{z_{ij}\}$. These are shown in equations (29-34), in Theorem 3 below. We note that these constraints are obtained independently in [26] and [23].

In addition, the variables $\{z_{ij}\}$ satisfy several other inequality constraints. Consider a station $\sigma \neq \sigma(j)$. It may or may not be working when $x_j(\tau_n) \geq 1$. Hence for $\sigma \neq \sigma(j)$, one only has,

$$x_j(\tau_n) \geq 1 \Rightarrow \sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n) \leq 1.$$

Hence

$$\sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n)x_j(\tau_n) \leq x_j(\tau_n). \quad (27)$$

Thus, upon taking expectations in (27), and using (25), we obtain the *non-idling inequality constraints* (35), shown in Theorem 3 below.

Also, clearly all the $z_{ij}$'s are non-negative, yielding (36).

Note finally that due to (25), the mean number in the system can be written in terms of the $z_{ij}$'s as,

$$E|x(\tau_n)| = \sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij}.$$

Thus we see that the mean number is given by a linear expression in the $z_{ij}$'s, which in turn are constrained by several linear equalities and inequalities. The situation is ripe for linear programming.

*Theorem 3* (Kumar and Kumar) Consider any non-idling stationary scheduling policy.
(i) In steady–state, the mean number of parts is bounded above by the value of the linear program:

$$\max \sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} \quad (28)$$

subject to

$$2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11} = 0 \quad (29)$$

$$2\mu_{j-1} z_{j-1,j} + 2\lambda - 2\mu_j z_{jj} = 0 \text{ for } j = 2, \dots, L \quad (30)$$

$$\lambda \sum_{\{j:\sigma(j)=\sigma(2)\}} z_{j2} - \lambda - \mu_1(z_{12} - z_{11}) - \mu_2 z_{21} = 0, \quad (31)$$

$$\lambda \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} - \mu_1 z_{1j} - \mu_j z_{j1} + \mu_{j-1} z_{j-1,1} = 0$$
$$\text{for } j = 3, \dots, L, \quad (32)$$

$$\mu_{i-1} z_{i-1,i+1} - \mu_i z_{i,i+1} - \lambda + \mu_i z_{ii} - \mu_{i+1} z_{i+1,i} = 0$$
$$\text{for } i = 2, \ldots, L-1, \quad (33)$$

$$\mu_{i-1} z_{i-1,j} - \mu_i z_{i,j} + \mu_{j-1} z_{j-1,i} - \mu_j z_{ji} = 0$$
$$\text{for } i = 2, \ldots, L-2, \text{ and } j = i+2, \ldots, L \quad (34)$$

$$\sum_{\{j : \sigma(j) = \sigma\}} z_{ji} \leq \sum_{\{j : \sigma(j) = \sigma(i)\}} z_{ji}$$
$$\text{for } i = 1, \ldots, L, \text{ and } \sigma \neq \sigma(i) \quad (35)$$

$$z_{ij} \geq 0 \text{ for } i = 1, \ldots, L \text{ and } j = 1, \ldots L. \quad (36)$$

(ii) The mean number of parts in steady-state is bounded below by the same linear program, with a "min" replacing the "max" in (28).

The bounds of Theorem 3 can be sharpened using policy-specific constraints. For example, consider a buffer priority policy using the ordering $\{b_{\theta(1)}, b_{\theta(2)}, \ldots, b_{\theta(L)}\}$. Clearly, if $b_i$ and $b_j$ share the same machine, and $b_i$ has higher priority than $b_j$, then

$$x_i(\tau_n) \geq 1 \quad \Rightarrow \quad w_j(\tau_n) = 0.$$

Hence

$$x_i(\tau_n) w_j(\tau_n) = 0.$$

Thus, we obtain the additional *buffer priority constraints*,

$$z_{ij} = 0 \text{ whenever } \sigma(i) = \sigma(j) \text{ and } \theta(i) < \theta(j).$$

We should note that for buffer priority policies, an upper bound as given in Theorem 3, with the buffer priority constraints appended to the LP, does *not* apply unless the total WIP under the policy has a finite first moment.

Example 2

Consider the system as in Figure 6, with $\lambda = 0.4$, $\mu_1 = \mu_3 = 1$, and $\mu_2 = 2$ as before. Consider the system operating under the LBFS policy as before. That is, buffer $b_3$ has pre-emptive priority over buffer $b_1$ and hence $z_{13} = 0$. Then using Theorem 3 we obtain an upper bound on the expected total number in the system of 3.8 units and a lower bound of 3.33 units. Thus, the maximum possible error in using the lower bound as an estimate for the actual expected total number in the system is $(3.8 - 3.33)/3.8$ or 12.28% of the true value.

We will conclude this section with a discussion on the tightness of the bounds obtained using the method described in this section. The efficacy of the bounds varies with several factors: the size and topology of the network being evaluated, the nature of the scheduling policy employed, the type of performance metric being estimated. There are cases when the bound is *exactly equal to the actual value*. An example of this is a single machine system with a static buffer priority rule. There are other cases when the bounds obtained, while not exact, are sufficiently

tight to allow one to compare policies. In [23], there is an example of a network in which the *upper* bound on the mean number of parts queued for processing under the Last Buffer First Serve rule is strictly smaller that the *lower* bound on the number of parts under the opposite First Buffer First Serve rule. Thus, the bounds serve as a useful guide to policy choice in this case, conclusively ruling out the choice of the First Buffer First Serve policy. The bounds obtained using this method can be improved upon in general by either using properties of the underlying Markov chains [27] or by increasing the moments considered in the analysis beyond quadratics [28]. Analytically tighter bounds can also be found in [29],[30].

## VII. Design: Scheduling of Re-Entrant Lines

Let us return to the problem of scheduling re-entrant lines, introduced earlier. The following results are drawn from [31].

One source of variability in wafer fabs is the variability in the flows. In particular, it is the variability in the time between consecutive arrivals to every station in the fab. We present a scheme for setting due dates that will simultaneously reduce burstiness of arrivals to each processing step, thus reducing variability in the flows. We do this by setting a due-date for reaching *each* processing step. Suppose $\lambda$ is the target throughput rate, i.e., the mean rate of release of new lots into the fab. For the $n$th lot being released into the fab, we can set the due date to reach step $k$ as $d_k(n) = \frac{n}{\lambda}$. Then, if we reduce the variance of the *lateness* in reaching step $k$, i.e., make lots uniformly early or late, we will reduce the burstiness of arrivals to step $k$[4]. Let us now turn to reducing the variance of lateness in reaching step $k$. Suppose $e_k(n)$ is the time at which the $n$th part arrives at step $k$. The lateness of the $n$-th lot in reaching step $l_k(n)$ is given by

$$l_k(n) \quad = \quad e_k(n) - d_k(n).$$

We will attempt to reduce the variance of lateness by implementing a variant of the least slack scheduling rule at each step $i$ where we define "slack" of the $n$th part in reaching step $k$ as

$$s_k(n) \quad = \quad d_k(n) - \zeta_{k,i}, \text{ where}$$

$\zeta_{k,i}$ is an *estimate* of the time remaining for a lot currently in step $i$ until it reaches buffer $k$. If $\zeta_{k,i}$ is accurate, this results in a "fair" policy that attempts to make all lots arriving at step $k$ equally early or late. We can also achieve the same results by implementing a least slack policy at each step $i$ with slack for the $n$th lot defined as

$$s_i(n) = \frac{n}{\lambda} - \zeta_i, \text{ where}$$

$\zeta_i$ is an estimate of the time remaining until exit from the system for a lot currently in step $i$. This version of the least slack policy is independent of the choice of the step

---

[4]Since we are only interested in minimizing the variance of lateness, there is no need to make the due-date depend on $k$.

$k$. If we have accurate estimates of the delay parameters $\zeta_i$, we hope to reduce the variability of arrivals to each step $k$ and thus reduce the consequent delays, and hence the average cycle time in the fab. This sequencing rule is called the *Fluctuation Smoothing Policy for Mean Cycle Time (FSMCT)*.

Using Little's law one can also show that the FSMCT policy is equivalent to a policy that at each time instant $t$ works on that class $j$ at station $\sigma = \sigma(j)$ such that

$$j = \arg \min_{\{k:\,\sigma(k)=\sigma\}} \sum_{i=k+1}^{L} (x_i(t) - \xi_i), \qquad (37)$$

where the thresholds $\xi_i \geq 0$ can be computed from the arrival rate and the delay estimates $\zeta_i$. These $\zeta_i$ can be computed by repeated simulation. That is, one first assumes $\zeta_i = 0$ and computes the resulting delays until exit. One then iterates with the new estimates of delays until one converges to a reasonable set of estimates of $\zeta_i$.

Regarding release policies, the *Workload Regulation Release* (WR) Policy is advocated in [32]. For simplicity we shall present the work load regulation policies in the setting of a fab with a single process and single product type. In a single bottleneck fab, one can chose to release work into the fab only when the total work that must be completed by the bottleneck in order to get rid of all of the current WIP in the system, is less than a threshold $A$. We shall call this policy $WR(A)$. The total work yet to be completed by the bottleneck $M$ can be calculated as

$$M \quad = \quad \sum_{i=1}^{L} M_i x_i,$$

where $L$ is the number of processing steps (which is equal to the number of buffers), $M_i$ is the amount of work to be done by the bottleneck on a part in processing step $i$ before it exits the system, and $x_i$ is the number of parts currently at processing step $i$ (i.e., buffer $b_i$). The $WR(A)$ policy then releases work into the system only when $M \leq A$. The choice of A determines the throughput rate that will be sustained under this release policy, and thus will have to be tuned to match the rate required to ensure that the backlog of orders does not grow without bound.

When there is more than one bottleneck in the fab, we can adapt the workload regulation policy described above for the one bottleneck case in many ways. First, we could just replace the workload $M$ by the *sum* of the workloads for each of the bottlenecks, and then pick a new threshold $A$ which reflects this as well. This approach does not differentiate between the bottlenecks and so the interactions between the bottlenecks are ignored. This may not be such a good idea. Alternatively, we could replace the single index $A$ by multiple indices where we explicitly track the workload for each of the machines and compare these against individual thresholds and release work into the system when any one of the workloads falls below its respective threshold.

We now present excerpts from a simulation case study of an R&D fab carried out first by Wein [32] and later by

TABLE I
CYCLE TIME PERFORMANCE COMPARISONS OF R & D FAB

| Policies | FIFO | SRPT | FSMCT |
|---|---|---|---|
| Deterministic | 261.67 | 280.34 | 234.97 |
| CONWIP | 301.59 | 297.43 | 271.12 |
| Workload Reg. | 253.93 | 273.35 | 229.66 |

Kumar et. al [31]. The fab has a single process comprising of 172 operations carried out at 24 stations, each consisting of one or more identical tools or machines. Many of these stations are visited more than once.

At this rate, the fab has one bottleneck, Station 14, which is utilized over 90% of the time. Three release policies described in the previous section are compared: deterministic release, the CONWIP release rule and the Workload regulation rule $WR(A)$ with $A$ being the threshold for the work at Station 14 below which additional wafers are released into the system. Both the CONWIP level and the threshold are chosen so as to achieve the target throughput.

The Fluctuation Smoothing Policy for Mean Cycle Time (FSMCT) is compared against the First In First out (FIFO) sequencing rule and the Shortest Expected Remaining Processing Time (SRPT) rule described in the previous section under each of the release policies described above. The performance metric used in the Mean Cycle Time of wafers in the fab. The results are tabulated in Table 1. A difference of 5 units is statistically significant and hence the combination of the Workload regulation policy and the FSMCT sequencing rule outperforms all other combinations of policies. The simulations in [31] affirm the choice of this release policy. While its mean cycle–time under FSMCT is almost the same as deterministic release, its variance is smaller. The improvements are substantial, and, as statistically tested in [31], significant.

## VIII. DESIGN: BROWNIAN APPROXIMATIONS

In the previous section, we presented a heuristic, simulation-based approach to policy design. In this section, we introduce an analytical method of policy design for queueing networks, due to Harrison [33]. We provide a concrete application of this method by rederiving the FSMCT policy for the reentrant line of Figure 6.

In this method, we begin by imagining a sequence of systems, indexed by $n$ in which a system parameter, such as the arrival rate, approaches a critical value. In our example, imagine a sequence of arrival rates $\lambda^n$ approaching the critical value $\lambda = 0.5$, where the nominal load at machine $\sigma_1$, is at the critical value $\rho_1 = 1$. The Brownian method is applicable in such *heavy traffic* conditions, when the nominal load at some of the resources is close to the critical value of 1. Next, we formally derive and solve a limiting Brownian control problem in terms of processes scaled as in a functional central limit theorem (e.g. queue lengths scaled as $x(n^2 \cdot t)/n$). We then interpret the solution to the Brownian control problem to derive a scheduling policy for the original network. Finally, we establish some optimality

properties of the derived policy in the asymptotic heavy traffic regime. See Harrison and Wein [34] and Kumar [35] for the complete sequence of these steps in the context of two-machine systems operated under CONWIP policies.

Consider the system of Figure 6 in the heavy traffic regime, with $\lambda = 0.5$ and $\mu_1 = \mu_3 = 1$ and $\mu_2 = 2$. We can argue that in the heavy traffic limit, queueing delays at buffer $b_2$ will be negligible in comparison to those at buffers $b_1$ and $b_3$. The system behaves as if server $\sigma_2$ and buffer $b_2$ did not exist at all.

For the reduced system we can write down the Brownian control problem as follows. Let $Z_i(t)$, $i = 1, 3$ denote the analogs of buffer levels in the Brownian model, corresponding to the formal limits of the scaled queue lengths in the original discrete system $x_i(n^2 \cdot t)/n$, and define the nominal workload by

$$W(t) = (\mu_1^{-1} + \mu_3^{-1})Z_1(t) + \mu_3^{-1}Z_3(t),$$

where $W(t)$ denotes the total work for server $\sigma_1$ in order to empty out the current contents of the system. In general we can decompose $W(t)$ as

$$W(t) = X(t) + Y(t),$$

where $X(t)$ is a Brownian motion with zero drift and variance $\sigma^2$, corresponding to the netput process (or the nominal difference between the cumulative input and output processes) and $Y(t)$ is a non-negative non-decreasing control process adapted to the Brownian motion $X(t)$ that cooresponds to the cumulative idleness process at server $\sigma_1$. We consider the objective of minimizing the long run average buffer levels using this control.

The complete Brownian control problem in the so-called equivalent Workload formulation (see Harrison and Wein [36] or Van Mieghem and Harrison [37] for example, for details) is the following.

$$\min_{\{Y(.)\}} \lim_{T \to \infty} \frac{E[\int_0^T (Z_1(t) + Z_3(t))\, dt]}{T} \quad (38)$$

$$W(t) = (\mu_1^{-1} + \mu_3^{-1})Z_1(t) + \mu_3^{-1}Z_3(t) \quad (39)$$

$$W(t) = X(t) + Y(t) \quad (40)$$

$$W(t) \geq 0 \quad (41)$$

$$Z_1(t) \geq 0 \quad (42)$$

$$Z_3(t) \geq 0 \quad (43)$$

The optimal solution to this problem can be derived by observing that

$$W(t) \geq X(t) - \inf_{\{0 \leq s \leq t\}} X(s),$$

for all $t$ if $W(t) \geq 0$ for all $t$, and hence $W(t)$ is minimized for every $t$ by choosing

$$Y(t) = -\inf_{\{0 \leq s \leq t\}} X(s). \quad (44)$$

See Chapter 2 of Harrison [38] for details. Moreover, from (39) the configuration that minimizes $Z_1(t) + Z_3(t)$ for every

$t$ is $Z_3(t) = 0$ and $Z_1(t) = W(t)/(\mu_1^{-1} + \mu_3^{-1})$. Since this solution minimizes $Z_1(t) + Z_3(t)$ for every $t$ given $X(t)$, this solution optimizes more than (38) and is called a *path-wise* solution.

The optimal solution can be interpreted as follows. Equation (44) says that $Y(t)$ does not increase unless $W(t) = 0$, and thus can be interpreted as saying that the optimal policy is non-idling. That is, unless $x_1(t) + x_3(t) = 0$ in the original unscaled discrete system, machine $\sigma_1$ is not allowed to idle. This, of course, is not a very useful policy specification since one expects any reasonable policy to be non-idling. One can however get much more insight by considering the specification that $Z_3(t) = 0$ at all times $t$. In light of the scaling $x(n^2 \cdot t)/n$, this can be interpreted as saying that one must strive to keep the contents of buffer $b_3$, $x_3(t) \leq \xi_3$, where $\xi_3$ is some fixed threshold in the original unscaled system . In order to achieve this, one gives priority to buffer $b_3$ whenever $b_3$ exceeds $\xi_3$. Relating this to (37) of the previous section, one can see that this is the FSMCT policy described there. Thus, we have derived the FSMCT policy by interpreting the optimal solution to the Brownian control problem.

## IX. CONCLUSIONS AND FUTURE DIRECTIONS

In this article, we have described the scheduling function in semiconductor wafer fabs, and identified the key trade-offs to be evaluated in designing scheduling policies. We have surveyed some of the sequencing rules and release policies used in semiconductor manufacturing. We have presented a queueing network model of fabs and discussed queueing network based analytical tools for both performance evaluation as well as dynamic control of such systems.

The model presented in this paper is deliberately kept simple in order to better serve the tutorial function. Most of the results illustrated here can be extended to situations with (a) multiple part types, each with their own process sequences, (b) probabilistic routing such as those arising from rework, (c) non-exponential distributions for processing times and interarrival times (using approximations based on the method of stages), and (d) systems with autonomous machine failures. Several other model extensions such as set-ups and changeovers are harder to handle and are discussed further below.

Regarding future directions, there are several ways in which the basic results presented in this paper have been and can continue to be extended. One direction in which such extensions occur is that of methodological improvement. For example, the Linear Programming based performance bounds of [23] can be strengthened to yield better bounds, see for example, [27], [29], [30]. In another direction, one could use the fluid model not only for stability analysis but also for policy design, see for example [39], [40]. A methodological lacuna in the Brownian approach is the difficulty with interpreting the solution of the Brownian control problem as a policy in the original network. Some progress has been made towards obtaining a systematic procedure for doing this [41].

Another direction in which the results presented here can be extended is improving the queueing network model itself. For example, one could explicitly model the set-up and change-over times. See [42], [43] for examples of such models and their implications. Systems with batch servers is yet another such example [44], [45]. Finally, there are problems such as workforce scheduling for which, to the best of our knowledge, there are no existing queueing network solutions that are both relevant and rigorous.

Queueing network models are universally employed in wafer fabs for simulation along the lines of Section 7. Most fabs run simulation models every day, and they are used not only operationally for seeing the impact of short term decisions but also for long term planning such as preventive maintenance scheduling and capacity addition. Analytical methods haven't yet taken root in industry. Reasons for this could include the paucity of definitive results, complexity of the methods, the need for data (such as distributional data) that is not easily available in actual fabs, and the lack of practical software tools based on these methods that integrate with the workflow management and ERP systems of the fabs. But, this remains a fertile area for future development, as pointed out by the International Technology Roadmap for Semiconductors put out by an international consortium that includes the US industry consortium Sematech. In the 1999 report, they emphasize in their factory integration section that "[w]hile each manufacturer must develop operations policies that fit its individual business model, additional theoretical development is needed to derive and analyze general operational policies [46]."

### References

[1] E. Brockmeyer, H. L. Halstrom, and A. Jensen, *The life and works of A. K. Erlang.* Academy of Technical Science, Copenhagen, 1948.

[2] J. R. Jackson, "Jobshop-like queueing systems," *Management Science*, vol. 10, pp. 131–142, Aug. 1963.

[3] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed and mixed networks of queues with different classes of customers," *J. Assoc. Comp. Mach.*, vol. 22, no. 2, pp. 248–260, 1975.

[4] D. P. Connors, G. E. Feigin, and D. D. Yao, "A queueing network model for semiconductor manufacturing," *IEEE Trans. Semiconductor Manufacturing*, vol. 9, pp. 412–427, 1996.

[5] S. B. Gershwin, *Manufacturing Systems Engineering.* Prentice Hall, 1994.

[6] Y. Dallery and S. B. Gershwin, "Manufacturing flow line systems: a review of models and analytical results," *Queueing Systems*, vol. 12, pp. 3-94, 1992.

[7] J. Kim, R. C. Leachman, "Dynamic Release Control Policy for Semiconductor Wafer Fabrication Lines," *J. Oper. Res. Soc.*, vol. 47, pp. 1516-1525, 1996.

[8] G. Liberopoulos and Y. Dallery, "A unified framework for pull control mechanisms in multi-stage manufacturing systems," *Ann. Oper. Res.*, vol. 93, pp. 325-355.

[9] R. W. Atherton and J. E. Dayhoff, "Signature Analysis: Simulation of Inventory, Cycle Time, and Throughput trade-offs in Wafer Fabrication," *IEEE Trans. Comp. Hyb. Manuf. Tech.* vol. CHMT-9, pp. 498-507, 1986.

[10] S. S. Panwalker and W. Iskander, "A survey of scheduling rules," *Oper. Res.*, vol. 25, pp. 45–61, Aug. 1977.

[11] W. J. Hopp and M. L. Spearman, *Factory Physics*, Second Edition, Irwin McGraw-Hill, Chicago, 2000.

[12] C. R. Glassey and M. Resende, "Closed-loop job release control for VLSI circuit manufacturing," *IEEE Trans. Semiconductor Manufacturing*, vol. 1, pp. 147–153, 1988.

[13] R. Uzsoy, C. Y. Lee, and L. A. Martin-Vega, "A review of production planning and scheduling models in the semiconductor industry, Part I," *IIE Trans.*, vol. 24, no. 4, pp. 47–60, 1992.

[14] A. N. Rybko and A. L. Stolyar, "Ergodicity of stochastic processes describing the operations of open queueing networks," *Problems of Information Transmission*, vol. 28, pp. 199–220, 1992.

[15] P. R. Kumar and T. I. Seidman, "Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems," *IEEE Trans. Aut. Control*, vol. 35, pp. 289–298, Mar. 1990.

[16] P. R. Kumar, "Re-entrant lines," *Queueing Systems*, vol. 13, pp. 87–110, 1993.

[17] S. Lippman, "Applying a new device in the optimization of exponential queueing systems," *Oper. Res.*, vol. 23, pp. 687–710, 1975.

[18] F. P. Kelly, *Reversibility and Stochastic Networks.* Wiley, 1979.

[19] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Trans. Aut. Control*, vol. 40, pp. 251–260, Feb. 1995.

[20] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Ann. Appl. Prob.*, vol. 5, pp. 49–77, 1995.

[21] J. G. Dai and G. Weiss, "Stability and instability of fluid models for certain re-entrant lines," *Math. Oper. Res.*, vol. 21, pp. 115–134, 1996.

[22] S. Kumar and P. R. Kumar, "Fluctuation smoothing policies are stable for stochastic re-entrant lines," *Discrete Event Dynamic Systems*, vol. 6, pp. 361–370, 1996.

[23] S. Kumar and P. R. Kumar, "Performance bounds for queueing networks and scheduling policies," *IEEE Trans. Aut. Control*, vol. 39, pp. 1600–1611, Dec. 1994.

[24] S. Kumar and P. R. Kumar, "Closed Queueing Networks in Heavy Traffic: Fluid Limits and Efficiency," in *Stochastic Networks: Stability and Rare Events*, Springer Lecture Notes in Statistics 117, pp. 41–64, 1996.

[25] P. R. Kumar and S. P. Meyn, "Duality and linear programs for stability and performance analysis of queuing networks and scheduling policies," *IEEE Trans. Aut. Control*, vol. 41, pp. 4–17, Jan. 1996.

[26] D. Bertsimas, I. C. Paschalidis, and J. N. Tsitsiklis, "Optimization of multiclass queueing networks:polyhedral and nonlinear characterizations of achievable performance," *Ann. Appl. Prob.*, vol. 4, no. 1, pp. 43–75, 1994.

[27] J. Morrison and P. R. Kumar, "New linear program performance bounds for queueing networks," *J. Opt. Th. Appl.*, vol. 100, pp. 575–597, Mar. 1999.

[28] S. Kumar, R. Srikant and P. R. Kumar, "Bounding blocking probabilities and throughput in queueing networks with buffer capacity constraints," *Queueing Systems,* vol. 28, pp.55-77, 1998.

[29] H. Jin, J. Ou, and P. R. Kumar, "The throughput of irreducible closed Markovian queueing networks: Functional bounds, asymptotic loss, efficiency, and the Harrison-Wein conjectures," *Math. of Oper. Res.*, vol. 22, pp. 886-920, 1997.

[30] C. Humes, Jr., J. Ou, and P. R. Kumar, "The delay of open Markovian queueing networks: Uniform functional bounds, heavy traffic pole multiplicities, and stability," *Math. of Oper. Res.*, vol. 22, pp. 921-954, 1997.

[31] S. C. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient

scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Trans. Semiconductor Manufacturing*, vol. 7, pp. 374–388, Aug. 1994.

[32] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Trans. Semiconductor Manufacturing*, vol. 1, pp. 115–130, Aug. 1988.

[33] J. M. Harrison, "Brownian models of queueing networks with heterogeneous customer populations," in *Stochastic Differential Systems, Stochastic Control Theory and Applications* (W. Fleming and P. L. Lions, eds.), vol. 10 of *Proceedings of the IMA*, pp. 147–186, Springer-Verlag, New York, 1988.

[34] J. M. Harrison and L. M. Wein, "Scheduling networks of queues: Heavy traffic analysis of a two-station closed network," *Oper. Res.*, vol. 38, pp. 1052–1064, Dec. 1990.

[35] S. Kumar, "Two-server closed networks in heavy traffic: Diffusion limits and asymptotic optimality," *Ann. Appl. Prob.*, 1999. Submitted.

[36] J. M. Harrison and L. M. Wein, "Scheduling network of queues: Heavy traffic analysis of a simple open network," *Queueing Systems*, vol. 5, pp. 265–280, 1989.

[37] J. M. Harrison and J. A. Van Mieghem, "Dynamic control of Brownian networks: State space collapse and equivalent workload formulations," *Ann. Appl. Prob.*, vol. 7, pp. 747–771, 1997.

[38] J. M. Harrison, *Brownian Motion and Stochastic Flow Systems*. Wiley, 1985.

[39] R.-R. Chen and S. P. Meyn, "Value iteration and optimization of multiclass queueing networks," *Queueing Systems*, vol. 32, pp. 65–97, 1999.

[40] F. Avram, D. Bertsimas, and M. Ricard, "Fluid models of sequencing problems in open queueing networks; an optimal control approach," in *Stochastic Networks* (F. Kelly and R. Williams, eds.), vol. 71 of *Proceedings of the IMA*, pp. 199–234, Springer-Verlag, New York, 1995.

[41] J. M. Harrison, "The BIGSTEP approach to flow management in stochastic processing networks," in *Stochastic Networks: Stochastic Control Theory and Applications* (F. Kelly, S. Zachary, and I. Ziedins, eds.), pp. 57–90, Oxford University Press, 1996.

[42] J. R. Perkins and P. R. Kumar, "Stable distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Trans. Aut. Control*, vol. 34, pp. 139–148, Feb. 1989.

[43] C. Chase and P. R. Ramadge, "On real-time scheduling policies for flexible manufacturing systems," *IEEE Trans. Aut. Control*, vol. 37, pp. 491–496, Apr. 1992.

[44] C. R. Glassey and W. W. Weng, "Dynamic batching heuristics for simultaneous processing," *IEEE Trans. Semiconductor Manufacturing*, vol. 4, pp. 77–82, 1991.

[45] R. K. Deb and R. F. Serfozo, "Optimal control of batch service queues," *Ann. Appl. Prob.*, vol. 5, pp. 340–361, 1973.

[46] Sematech, "International Technology Roadmap for Semiconductors," pp.192, 1999.